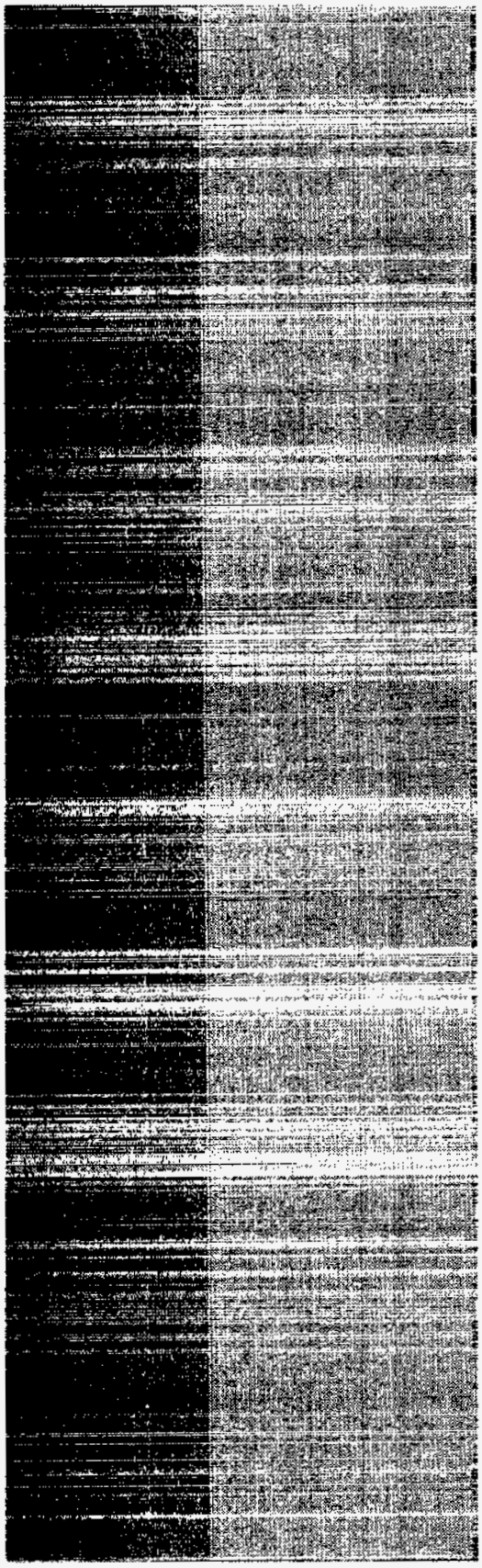
LA-13514-T
Thesis



Development of Monteburns: A Code That Links MCNP and ORIGEN2 in an Automated Fashion for Burnup Calculations

Los Alamos
NATIONAL LABORATORY

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36.

This thesis was accepted by the Department of Chemical and Nuclear Engineering, the University of New Mexico, Albuquerque, New Mexico, in partial fulfillment of the requirements for the degree of Master of Science. The text and illustrations are the independent work of the author and only the front matter has been edited by the CIC-1 Writing and Editing Staff to conform with Department of Energy and Los Alamos National Laboratory publication policies.

An Affirmative Action/Equal Opportunity Employer

LA-13514-T
Thesis

UC-700 and UC-714
Issued: December 1998

Development of Monteburns: A Code That Links MCNP and ORIGEN2 in an Automated Fashion for Burnup Calculations

Holly R. Trellue

Los Alamos
NATIONAL LABORATORY
Los Alamos, New Mexico 87545

ACKNOWLEDGMENTS

First, I sincerely thank everybody in the Nuclear Systems Design and Analysis Group (TSA-10) at Los Alamos National Laboratory. I especially thank Dave Poston for coming up with this idea and for all your enormous help and knowledge - you've taught me so much! John Buksa and Stacey Eaton, thank you for supporting me and allowing me the freedom to work on this research. Thanks also to Mike Houts and Paul Chodak for your technical advice and support. Additionally, I send a "thank you" to Deborah Bennett and Al Neuls for taking the time to care about the students in your group!

Next, I acknowledge Larry Sanchez at Sandia National Laboratories for introducing me to the world of FORTRAN77 programming, MCNP, and ORIGEN2, as well as for all the knowledge you gave me. I also thank my committee members at UNM, Dr. Norm Roderick, Dr. Robert Busch, and Dr. Gary Cooper, for taking the time to read this document and for everything you taught me both as a undergraduate and a graduate student.

Finally, I thank my parents, Judy Emmett and Ron and Patricia Trellue, and my best friend, Rosa Canchucaja, for always being there for me, and my boyfriend Dave Fuehne for all your encouragement and support. I also have to thank my dog girls, Cleopatra and Penelope, for your comic relief and doggie kisses - don't worry - now I'll have more time to give you belly rubs and biscuits!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Development of *Monteburns*: A Code That Links MCNP and ORIGEN2 in an Automated Fashion for Burnup Calculations

by

Holly R. Trellue

ABSTRACT

Monteburns is a fully automated tool that links the Monte Carlo transport code MCNP with the radioactive decay and burnup code ORIGEN2. *Monteburns* produces many criticality and burnup computational parameters based on material feed/removal specifications, power(s), and time intervals. This code processes input from the user indicating the system geometry, initial material compositions, feed/removal, and other code-specific parameters. Results from MCNP, ORIGEN2, and other calculations are then output successively as the code runs. The principle function of *monteburns* is to first transfer one-group cross sections and fluxes from MCNP to ORIGEN2, and then transfer the resulting material compositions (after irradiation and/or decay) from ORIGEN2 back to MCNP in a repeated, cyclic fashion. The main requirement of the code is that the user have a working MCNP input file and other input parameters; all interaction with ORIGEN2 and other calculations are performed by *monteburns*.

This report presents the results obtained from the benchmarking of *monteburns* to measured and previously obtained data from traditional Light Water Reactor systems. The majority of the differences seen between the two were less than five percent. These

were primarily a result of variances in cross sections between MCNP, cross section libraries used by other codes, and observed values. With this understanding, this code can now be used with confidence for burnup calculations in three-dimensional systems. It was designed for use in the Accelerator Transmutation of Waste project at Los Alamos National Laboratory but is also being applied to the analysis of isotopic production/destruction of transuranic actinides in a reactor system. The code has now been shown to sufficiently support these calculations.

1.0 INTRODUCTION

The past few decades have brought growth in a number of areas, two of which include the nuclear industry and computer technology. As restrictions placed upon and costs involved with experimental facilities increase (due to environmental and radiological health concerns), the value of computer modeling also increases. It has become possible to model various types of nuclear systems (including full reactor cores) and perform complex decay and burnup calculations in a matter of seconds. With the increase in computer technology, the number of computer codes available to perform nuclear-related calculations has increased, and often the user wants to run two or more codes concurrently. Thus, many linkage codes have been written to allow concurrent use of these "main" codes in an automated fashion. Two popular codes used in the design of nuclear systems are MCNP™ and ORIGEN2, and the code presented in this report is a linkage code for these two "main" codes.[1]

MCNP (Monte Carlo N-Particle transport code) is widely used to perform Monte Carlo calculations of neutron, photon, and/or electron transport.[1] MCNP is primarily used for analyzing the exact geometry and material composition of a system to determine the behavior of particles in that system (see Section 2.1 for a more detailed description of MCNP). It cannot, however, determine the effect that irradiation (burnup) has on the materials within the system (i.e., radioactive decay and burnup calculations). Instead, this is the function of the code ORIGEN2 (The Oak Ridge National Laboratory (ORNL) Isotope Generation and Depletion Code), which analyzes the burnup and concurrent decay of isotopes in a system over time.[2] The limitation of ORIGEN2 is that it does not take into account the geometry of a system. The geometry, among other things, influences cross sections and neutron fluxes at various positions in the material/region(s) being analyzed. These geometry-dependent parameters of the system

---

[1] Radiation Safety Information Computational Center (RSICC) Code Packages CCC-660 and CCC-371.

can be determined by MCNP. Thus, it is desirable to link MCNP and ORIGEN2 to allow accurate calculations of spatial isotope generation and depletion in a physical system.

The basis for the work presented in this paper is the need for a fully automated linkage code that transfers material compositions and cross sections for any three-dimensional (3-D) system from MCNP to ORIGEN2, transfers the materials remaining after irradiation from ORIGEN2 to MCNP, obtains new cross sections, criticality parameters, and flux/energy spectrums from MCNP, and then transfers materials back to ORIGEN2 in a cyclic fashion for as many time steps as needed. Additionally, three other features related to overall performance were desired: 1) the option to irradiate more than one material as separate ORIGEN2 analyses from a single MCNP output file and combine them again after irradiation into a single MCNP input file, 2) the ability to transfer material from one region in MCNP to another, and 3) the capability to add or remove specified materials after each step in an automated fashion.

Initially, *monteburns* was specifically developed for use in the Accelerator Transmutation of Waste (ATW) project[3] because it could combine a detailed 3-D system model with burnup calculations in an automated fashion. The goal of the ATW project is to reduce the radiotoxicity of nuclear waste so that the radiotoxicity of ATW-treated waste after 300 years is less than that of untreated waste after 100,000 years (see Section 5.1 for more information). For this project, it is desired to have a linkage code that allows addition (referred to as "feed" in this document) and/or removal of material either continuously or discretely (all at one time). In addition, the code must be capable of burning more than one material region in ORIGEN2 and of combining isotopic compositions for each material into one main MCNP input file for a series of burnup steps. For ATW, all of these functions are performed and regions of spent fuel are rotated from the outside to the inside of the system to allow different amounts of

irradiation to occur in each. The code was also designed so that it can be used for reactor systems, as shown in Sections 4.1 and 5.2.

The name *monteburns* was chosen because it is a **Monte** Carlo **burn**up tool. The purpose of this document is threefold: 1) to present information relevant to the development of *monteburns* (i.e., background/previous work, theory and calculations used in the code), 2) to display results of benchmark calculations used to verify the performance of *monteburns* and of statistical analyses for several input parameters, and 3) to show current and future applications of *monteburns*.

2.0 BACKGROUND

Over the past few decades, the development of numerous computer codes has increased the utilization of computer modeling in solving nuclear design problems. For example, Los Alamos National Laboratory developed a Monte Carlo code, MCNP, which is used to model particle transport in a variety of nuclear systems. In addition, Oak Ridge National Laboratory designed a number of codes, including ORIGEN2, the radioactive decay and burnup code discussed in this document, and the SCALE package, which is a "Modular Code System for Performing Standardized Computer Analyses for Licensing Evaluation."[2] The SCALE package encompasses a variety of codes, including several (i.e., MORSE and KENO) that perform Monte Carlo transport calculations, and ORIGEN-S, which performs radioactive decay and burnup calculations (ORIGEN-S is a "newer" version of ORIGEN2). Concurrently, many commercial nuclear companies (both in the United States (US) and Europe), developed their own methods/codes for analyzing the effects of burnup on a reactor core. Many of these methods have been used and tested extensively, but many are not publicly available.

There have also been several codes written to link MCNP and ORIGEN2, some of which are discussed in Section 2.3. However, each of these linkage codes appears to have been developed for specific purposes and thus has certain limitations. *Monteburns* was developed to be as versatile as possible so that it can be applied to a large number of situations and give the user a variety of choices of operational parameters while simplifying required user training.

Descriptions of the two codes linked by *monteburns*, MCNP and ORIGEN2, are included below, followed by a discussion of previously developed burnup codes. One of the main assumptions made by *monteburns* is that MCNP and ORIGEN2 perform

---

[2] Radiation Safety Information Computational Center (RSICC) Code Package CCC-545.

calculations well; benchmarking of them has already been performed, so no additional benchmarking is necessary.

2.1 MCNP

MCNP is a transport code that uses the Monte Carlo technique. The Monte Carlo technique is a statistical method in which estimations for particle characteristics are obtained through multiple computer simulations of the behavior of individual particles in a system. The probability that a particle behaves in a certain manner (scatters, absorbs, fissions) is obtained from the cross sections for the material(s) with which the particle interacts. For example, if a material is a pure absorber, the probability that a particle interacting with this material is absorbed is 100%. If the material is both an absorber and a scatterer, then the probability of absorption is equal to the ratio of the absorption cross section to the total cross section (absorption plus scatter). It follows that the probability of scatter is equal to the ratio of the scattering cross section to the total cross section. After a particle has undergone a scatter, it remains in the system to undergo another interaction. A Monte Carlo code keeps track of the position of each particle before and after it scatters and/or is absorbed, as well as any neutrons produced from fission interactions. If a particle travels outside of the system, then it is considered to have "leaked." At the end of the "life" of the particle, it either leaks from the system or is absorbed in a material. In the case of a neutron being absorbed in fissile material and causing a fission, the location and number of new neutrons created is recorded.

A Monte Carlo code generates a statistical history for a particle based on random samples from probability distributions used in calculations to determine 1) the type of interaction the particle undergoes at each point in its life, 2) the resulting energy of the particle if it scatters, and/or 3) the number of neutrons it produces if it causes a fission. Thus, a Monte Carlo code models the series of events that occur in the lives of a large number of particles to determine the flux of different types of particles in various

locations in the system. The particles of the most interest in criticality/burnup calculations are neutrons because they are the ones that interact with fissile materials to produce energy as well as more neutrons.

MCNP is used to model the events in the lives of neutrons, photons, and/or electrons. The cross sections for the particles are obtained from numerous material cross section libraries containing a number of isotopes at various operating temperatures. MCNP uses these libraries in a continuous-energy fashion, which means that it obtains the specific cross section for a given energy rather than looking at grouped cross section sets, in which the cross sections represent an average of a particular range of energies.

MCNP can also calculate the effective multiplication factor (k_{eff}) for a system, which is the number of neutrons produced in one generation divided by the number of neutrons that existed in the previous generation, indicating how close the system is to being critical (k_{eff} of 1.0). Table 1 shows the condition of a system at various values of k_{eff}. A reactor is typically operated at a k_{eff} around 1.0 as the system is self-sustaining at that point (i.e., requires no new source of neutrons).

MCNP is a valuable tool in that it helps to design a system to operate at a certain condition. MCNP was developed by personnel at Los Alamos National Laboratory (LANL), serves a large number of government and institutional organizations, and has been well maintained and updated. For more information about Monte Carlo codes or MCNP in particular, see Ref. 1 or 5.

Table 1. Conditions of k_{eff}

Value of k_{eff}	Condition [4]
$k_{eff} < 1.0$	Subcritical
$k_{eff} = 1.0$	Critical
$k_{eff} > 1.0$	Supercritical

2.2 ORIGEN2

ORIGEN2 is a version of the ORIGEN computer code, which is an isotope generation and depletion code used for performing radioactive decay and burnup analyses for a material. ORIGEN calculates the concentration of nuclides at numerous points throughout a decay or irradiation primarily using the matrix exponential method of equation solving. ORIGEN treats the full isotopic matrix of materials generated through irradiation by considering time-dependent formulation, destruction, and decay concurrently. The main calculation performed by ORIGEN is shown in Equation 1.[6]

$$\frac{dN_i}{dt} = \sum_j \gamma_{ji}\sigma_{f,j}N_j\phi + \sigma_{c,i-1}N_{i-1}\phi + \lambda_i' N_i' - \sigma_{f,i}N_i\phi - \sigma_{c,i}N_i\phi - \lambda_i N_i \qquad (1)$$

where: $\frac{dN_i}{dt}$ = change in concentration of nuclide i with time =

Formation rate - Destruction rate - Decay rate

Formation terms:

$\sum_j \gamma_{ji}\sigma_{f,j}N_j\phi$ = fission yield rate of N_i from fissionable nuclides N_j

$\sigma_{c,i-1}N_{i-1}\phi$ = transmutation rate of N_{i-1} into N_i by neutron capture

$\lambda_i' N_i'$ = radioactive decay rate of N_i' into N_i

Destruction terms:

$\sigma_{f,i}N_i\phi$ = fission rate of nuclide N_i

$\sigma_{c,i}N_i\phi$ = capture rate of nuclide N_i - (n,γ),(n,α),(n,p),(n,2n), and (n,3n)

Decay term:

$\lambda_i N_i$ = radioactive decay rate of nuclide N_i

where: γ_{ji} = fission yield of nuclide i from nuclide j (obtained from libraries)

$\sigma_{f,j}$ = microscopic fission cross section of nuclide j (cm^2 - from libraries)

N_j = concentration of nuclide j (gram-atoms - calculated)

ϕ = neutron flux in system (n/cm^2-s - input)

$\sigma_{c,i-1}$ = microscopic capture cross section of nuclide i-1 (cm^2 - from libraries)

λ_i = decay constant of nuclide i (s^{-1} - obtained from decay library)

The matrix exponential method used to solve this problem with a spectrum-averaged flux and one-group cross sections is shown in Equations 2 and 3.

$$\dot{N} = AN \tag{2}$$

$$N = N_o e^{At} \tag{3}$$

where: $\dot{N}$ = change of nuclide concentration with time

A = transition matrix with rate coefficients (decay, absorption, fission)

N = vector of nuclide concentrations at time t

N_o = vector of initial nuclide concentrations

The equation is then solved by obtaining a series expansion for the term e^{At}.

$$e^{At} = \sum_{m=0}^{\infty} \frac{(At)^m}{m!} \tag{4}$$

Sometimes difficulties occur in generating accurate values using the matrix exponential method, and either the Bateman equations[7] or the Gauss-Seidel iterative technique[8] is applied. The number of nuclides removed from the transition matrix and processed using the Bateman nuclide chain equations are determined by how many have half-lives (both absorption and fission) less than 10% of the time interval being investigated. Thus, having a shorter time interval in ORIGEN allows the Bateman

equations to be used in solving for the concentrations of a larger number of isotopes (as discussed in Section 3.3). This can be advantageous in that it often allows more accurate results to be obtained.

The input required for ORIGEN2 consists of three parts: cross section libraries, information about each decay/irradiation step, and initial material compositions. First, ORIGEN2 contains over 40 different data sets with one-group cross sections for various energy/system spectra. The user must decide which one to use, and transfer both the ORIGEN2 decay library and that cross section library to a file that can be read by ORIGEN2 (typically called *fort.9*). He/she must then enter identifiers for these libraries in the main ORIGEN2 input file. Second, this main ORIGEN2 input file must also contain detailed information required to run the code, including the length(s) of each decay and/or irradiation, the flux or power associated with each irradiation, and a description of what output parameters (and units of these parameters) are desired. Finally, the initial composition of the material being irradiated must be entered. This can either be part of the main ORIGEN2 input file, or it can be self-contained in its own file (usually called *fort.4*). The output for ORIGEN2 includes cross sections and fission yields used by the code as well as nuclide concentrations at each time step as specified by the user.

2.3 Previous Work

There are two main classes of codes that can be used to perform criticality calculations for nuclear systems: a Monte Carlo code, and a deterministic code. Monte Carlo techniques typically produce a statistical approximation of the answer for the exact geometry of the system, whereas deterministic codes numerically produce an exact solution of the diffusion and/or transport equations for the problem as modeled. Deterministic codes generally cannot solve such equations easily for complex geometries, so approximations on the geometry must be made.[5] Additionally, deterministic codes generally utilize less accurate cross section data (i.e., grouped versus continuous). With a

Monte Carlo code such as MCNP, a supplemental code, such as ORIGEN2, must be used to perform burnup analyses, and another code (i.e., a linkage code) is needed to interact between the two. Examples of such linkage codes include MOCUP,[9] COUPLE,[10] and SCAMP,[11] which are further discussed in the following sections and are compared in Table 2.

Table 2. Comparison of Linkage and/or Burnup Codes

Code	Description/Comparison	Ref
MOCUP[3]	Includes Monte Carlo, 3-D techniques and system-dependent parameters Links MCNP and ORIGEN2 with existing input files for each Modifies reaction rates, fluxes, and cross sections in ORIGEN2 Modifies nuclide compositions in MCNP after one burnup period	9
SCALE/ COUPLE	Allows Monte Carlo 3-D modeling and system-dependent parameters Develops multi-group cross sections and neutron fluxes for ORIGEN-S Modifies cross sections and fluxes at each time step Is a fully automated suite of programs and requires detailed training	10 6
SCAMP	Links MCNP and ORIGEN-S for burnup calculations of LWRs** Transfers material compositions after burnup to MCNP Does not transfer cross sections or fluxes	11
HELIOS CASMO	Performs transport calculations for a two-dimensional (2-D) geometry Couples subcomponents to perform fast, efficient calculations Uses multi-group ENDF* cross section libraries Does not include system-dependent axial effects	12 11
ANDROMEDA[4]	Performs one-dimensional diffusion calculations for fast reactors	-
DANTSYS[5]/ ORIGEN2	Calculates criticality parameters using transport theory Multi-group cross sections must be collapsed to one-group for ORIGEN2 Can perform detailed 3-D geometry calculations, but only with difficulty	13 2

* The United States Evaluated Nuclear Data Files, particularly ENDF/B-V or ENDF/B-VI versions[14]
** Light Water Reactors

[3] Radiation Safety Information Computational Center (RSICC) Code Package PSR-365.
[4] http://www.nea.fr/abs/html/nea-0321.html
[5] Radiation Safety Information Computational Center (RSICC) Code Package CCC-547.

In contrast, many deterministic codes used by the commercial nuclear industry (for example, HELIOS[12] and ANDROMEDA4) actually incorporate burnup as well as criticality calculations. These codes are designed for one- or two-dimensional lattice geometries and are often large, complex programs to execute. The other way to use a deterministic code that does not perform burnup calculations (for example, the Diffusion Accelerated Neutral Particle Transport System (DANTSYS) suite of codes)[13] is to link it with a burnup code such as ORIGEN2. Although deterministic codes can perform burnup calculations, they do not have the physical accuracy associated with a Monte Carlo code that models a detailed, 3-D geometry. These two categories of codes are discussed in the following sections with examples of each, but these only represent a small sample of the codes that have been written for burnup analyses; there are most likely other types of codes not presented here.

2.3.1 Linkage Codes

MOCUP (MCNP-ORIGEN2 Coupled Utility Program) is a MCNP/ORIGEN linkage code designed to transfer fluxes, reaction rates, nuclides, and cross sections from MCNP to ORIGEN2 using a number of user-supplied skeleton ORIGEN2 files, which are then modified with MCNP results. Then it extracts nuclide compositions from the ORIGEN2 output files and converts them into number densities, which are placed back into MCNP. However, it requires a certain structure for the initial MCNP input file (with comments indicating different locations in the file) and requires the user to create skeleton ORIGEN2 input files. It does not interact in an automated fashion with MCNP and ORIGEN2 for more than one time step; instead, the user must run each time step manually, adding feed materials, removing waste, and/or rotating regions. Although the MOCUP utility can be very useful for simple analyses involving MCNP and ORIGEN2, it does not work well with repeated structures, multi-materials, or the other limitations discussed previously.

COUPLE is one of the many modules that exist in the SCALE (Standardized Computer Analyses for Licensing Evaluation) suite of programs. The purpose of COUPLE is to produce multi-group cross section libraries from the ENDF data base and multi-group neutron fluxes, which are required as input for ORIGEN-S, from a detailed model of the system (typically developed using the SCALE module KENO). This program, along with other modules in SCALE (such as NITAWL, BONAMI, and/or XSDRNPM), allows system-dependent design characteristics (such as operating parameters and material compositions) to influence multi-group cross sections. This system is fully automated with the feature that a large suite of programs are used to represent a system as accurately as possible. Unfortunately, although these modules offer a number of options for performing calculations, they also require extensive, detailed training to execute properly.

SCAMP (SCALE-to-MCNP Post Processor) was a code written to link ORIGEN-S and MCNP for Pressurized-Water-Reactor (PWR) fuel assembly configurations. It transfers actinide and fission product compositions from the SCALE module ORIGEN-S to MCNP. However, it does not perform automated calculations for numerous steps or generate spectrum-averaged cross sections from MCNP to ORIGEN-S. The advantage of this program is that ORIGEN-S uses cross sections representative of typical PWR systems, whereas the data base for ORIGEN2 may not be as representative.

2.3.2 Discrete Ordinate Burnup Codes

There are a number of discrete ordinate burnup codes used in the commercial nuclear industry for analyzing the components of a nuclear reactor during operation. Two such examples are HELIOS and ANDROMEDA.

HELIOS performs neutron and gamma transport and burnup calculations for two-dimensional lattice geometries. It consists of three different processors: the main program, a pre-processor, and a post-processor. It was developed by Scandpower A/S

as a two-dimensional collision probability-based transport code. The associated HELIOS libraries are 34-energy group libraries based upon ENDF/B-VI data for a variety of temperatures. HELIOS is useful for performing quick calculations for various reactor physics constants but needs to be coupled with another code to obtain temperature coefficients or to model 3-D system effects. HELIOS is also fairly expensive to obtain. Additionally, CASMO, another widely used burnup code in the US commercial nuclear industry, performs calculations fairly similar to HELIOS.[12]

ANDROMEDA is a one-dimensional multi-group diffusion-burnup code developed in the Netherlands for use with fast reactor systems. The code is designed primarily for fuel-cycle analysis of fast breeder reactors by calculating regular and adjoint fluxes, material bucklings, kinetics parameters, material (fuel or poison) concentrations, and region dimensions at various steps throughout irradiation. ANDROMEDA collapses multi-group cross sections to several groups and analyzes cylindrical, spherical, and/or slab geometries. A variety of multi-group cross section libraries for ANDROMEDA are available.

3.0 DESCRIPTION OF CODE/THEORY

Although the linkage and burnup codes discussed in the previous section perform adequate calculations for the irradiation of materials in a system, they do not provide the entire range of parameters and functions useful in advanced nuclear burnup problems. For ATW and certain reactor systems (see Section 5), it is desired to have a code that performs automated burnup calculations for a 3-D system for more than one time step. It is also desirable to calculate spectrum-averaged cross sections and fluxes for each of these burnup steps. The Monte Carlo code MCNP was chosen to model the system because it is widely known and is capable of modeling in three dimensions as well as calculating spectrum-averaged cross sections and fluxes in different regions of the system. The code ORIGEN2 was chosen to perform calculations involving the change of nuclide concentrations because it is a stand-alone radioactive decay and burnup code with the characteristic that cross sections and material compositions can each be contained within separate input files, making them easy to modify for numerous burn steps.

In addition, it is preferred to have a linkage code involving little interaction with ORIGEN2 and with the ability to work with any MCNP input file (i.e., no format requirements for an ORIGEN2 or MCNP input file) without requiring detailed training. Other desired features include the ability to add and/or remove certain materials in a system at different burn steps, burn more than one material from the initial MCNP input file, and rotate materials from one region in the system to another. None of the linkage codes presented in Section 2.3.1 exhibit all of these options, and the deterministic codes in Section 2.3.2 do not analyze detailed, 3-D systems easily. Thus, the linkage code *monteburns* was designed to model the system accurately, incorporate all desired features, and make the input and training requirements as simple as possible. This section includes a brief description of the code, presents the calculations it performs, and describes the input required by and the output produced by *monteburns*.

3.1 Description of *Monteburns*

Monteburns is a UNIX c-shell command file (see Appendix A) that frequently interacts with a FORTRAN77[15] program, *monteb.f*, (see Appendix B) to produce criticality and burnup results based on material feed/removal specifications, power(s), and time intervals. Figure 1 shows how *monteburns* interacts with MCNP and ORIGEN2.

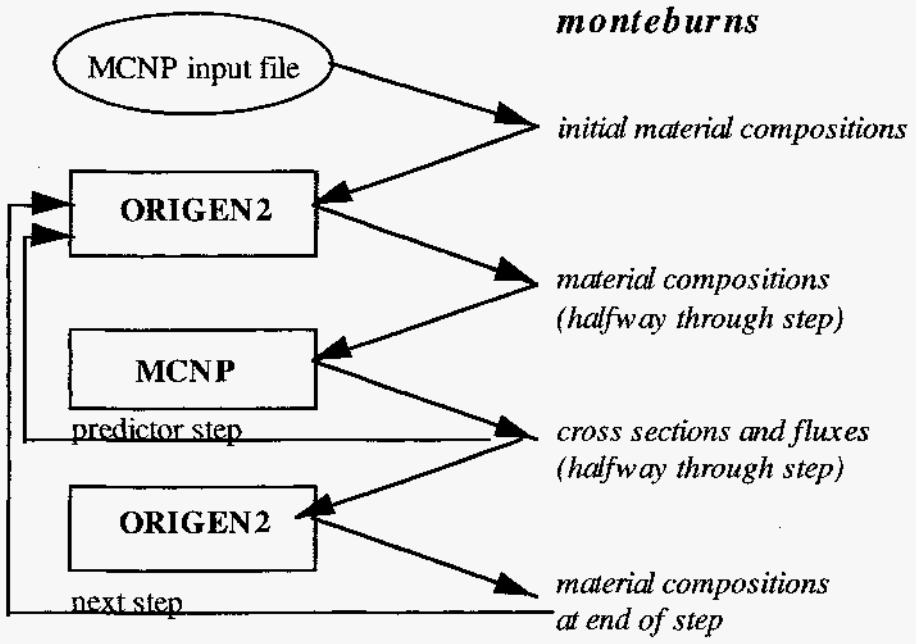


Figure 1. Interaction of *Monteburns* with MCNP and ORIGEN2

The primary way in which MCNP and ORIGEN2 interact through *monteburns* is that MCNP provides spectrum-averaged one-group microscopic cross sections and fluxes required for ORIGEN2, and ORIGEN2 provides material compositions halfway through and at the end of each irradiation step. These calculations may occur more than once throughout an irradiation period to obtain the best representation for a particular burn step (see Section 3.3.2 for more information about predictor steps).

Monteburns acts as a post-processor for MCNP and a pre- and post-processor for ORIGEN2. For each irradiation step, MCNP is run with material compositions halfway through the step (obtained from ORIGEN2), and relevant parameters are extracted by *monteburns* and input into ORIGEN2. A majority of information desired by the user is contained in the *monteburns* output (see Section 3.4), and additional information can be obtained in the future if desired (see Section 6). Nonetheless, *monteburns* was designed to eliminate the user's need to search through MCNP output files for results.

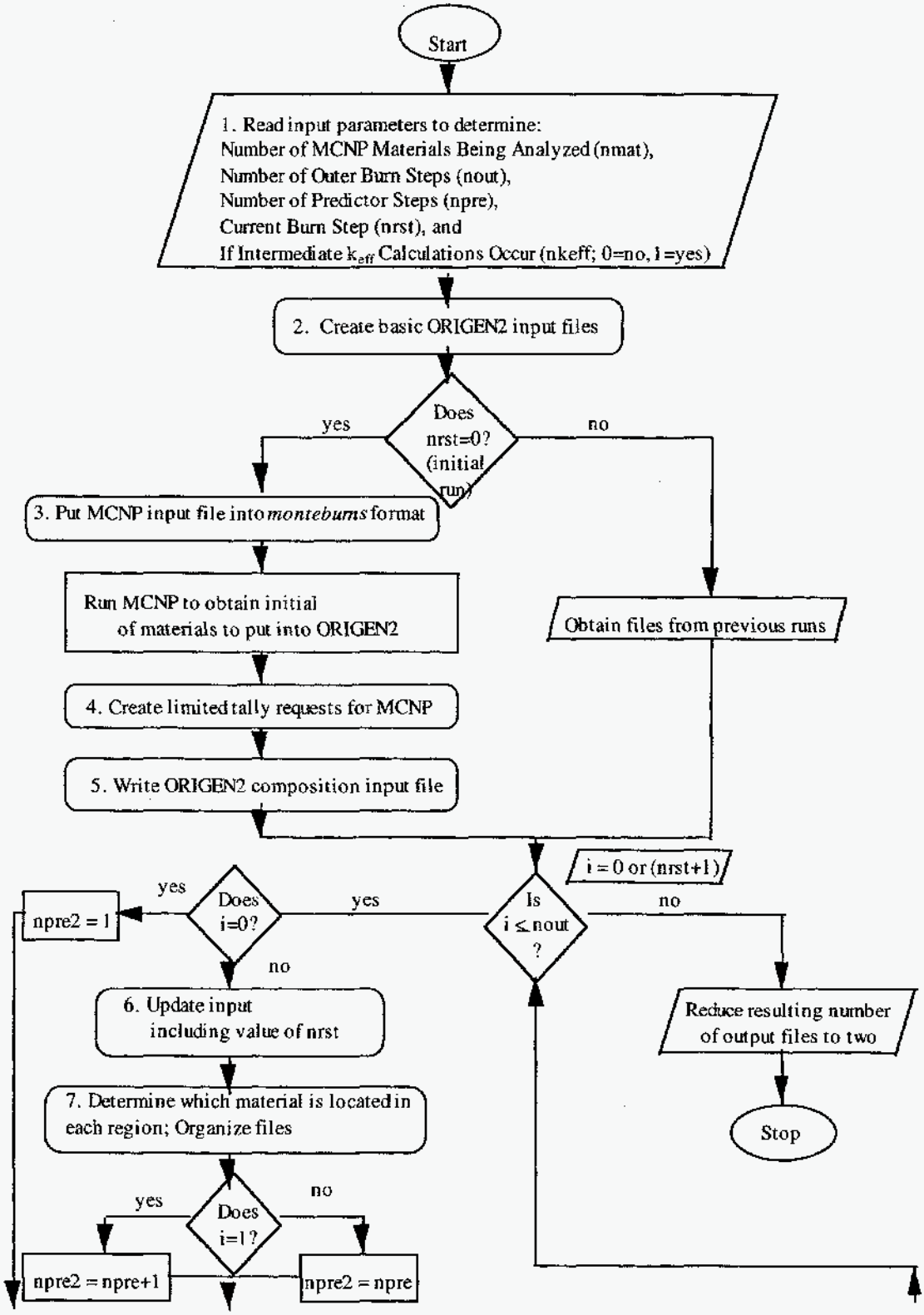
In addition, input files for ORIGEN2 are complex to write, and output files generated by ORIGEN2 are bulky and complicated to read. Thus, *monteburns* eliminates the user's need to create his/her own ORIGEN2 input files and to extract information from ORIGEN2 output files. *Monteburns* provides a file with cross section and decay libraries (*fort.9*), a material composition input file (*fort.4*), and a main ORIGEN2 input file (*mbori*), which contains commands as well as some feed and removal information (optional). All three of these files are created by *monteburns* for each material, and they provide all the information needed to execute ORIGEN2.

The FORTRAN77 program, *monteb.f*, which interacts with the c-shell file *monteburns*, consists of fifteen different parts, each of which performs a different function. These functions are displayed in the detailed flow chart of the c-shell file *monteburns* in Figure 2, where the numbers correspond to the list below.

1. read input parameters,
2. create basic ORIGEN2 input files for each main burn step based on continuous feed/removal information,
3. put the user's MCNP input file into *monteburns* format,
4. create tally requests for MCNP,
5. write ORIGEN2 composition input file, separating natural elements into individual isotopes,

6. update the *monteburns* input file to indicate the current step number and to update the list of isotopes being tracked,
7. determine which material is located in each region,
8. add discrete feed to ORIGEN2 composition input file (if requested by the user),
9. modify the previous MCNP input file with new material compositions,
10. modify ORIGEN2 input files for predictor steps to calculate compositions halfway through each burn step,
11. modify ORIGEN2 libraries with cross sections calculated by MCNP and ORIGEN2 input files with fluxes from MCNP,
12. calculate the recoverable energy per fission based on the actinide distribution,
13. perform discrete removal in the ORIGEN2 composition input file,
14. output results of ORIGEN2, and
15. calculate the amount of material burned and produced based on feed and inventory information.

The full range of calculations performed by *monteburns* is presented in Section 3.2, detailed input requirements are described in Section 3.3, and the results currently output by *monteburns* are displayed in Section 3.4.


Start
1. Read input parameters to determine:
Number of MCNP Materials Being Analyzed (nmat),
Number of Outer Burn Steps (nout),
Number of Predictor Steps (npre),
Current Burn Step (nrst), and
If Intermediate k_{eff} Calculations Occur (nkeff; 0=no, 1=yes)
2. Create basic ORIGEN2 input files
Does nrst=0? (initial run)
yes
no
3. Put MCNP input file into *montebums* format
Run MCNP to obtain initial of materials to put into ORIGEN2
4. Create limited tally requests for MCNP
5. Write ORIGEN2 composition input file
Obtain files from previous runs
i = 0 or (nrst+1)
Is i ≤ nout ?
yes
no
Does i=0?
yes
no
npre2 = 1
6. Update input including value of nrst
7. Determine which material is located in each region; Organize files
Reduce resulting number of output files to two
Stop
Does i=1?
yes
no
npre2 = npre+1
npre2 = npre

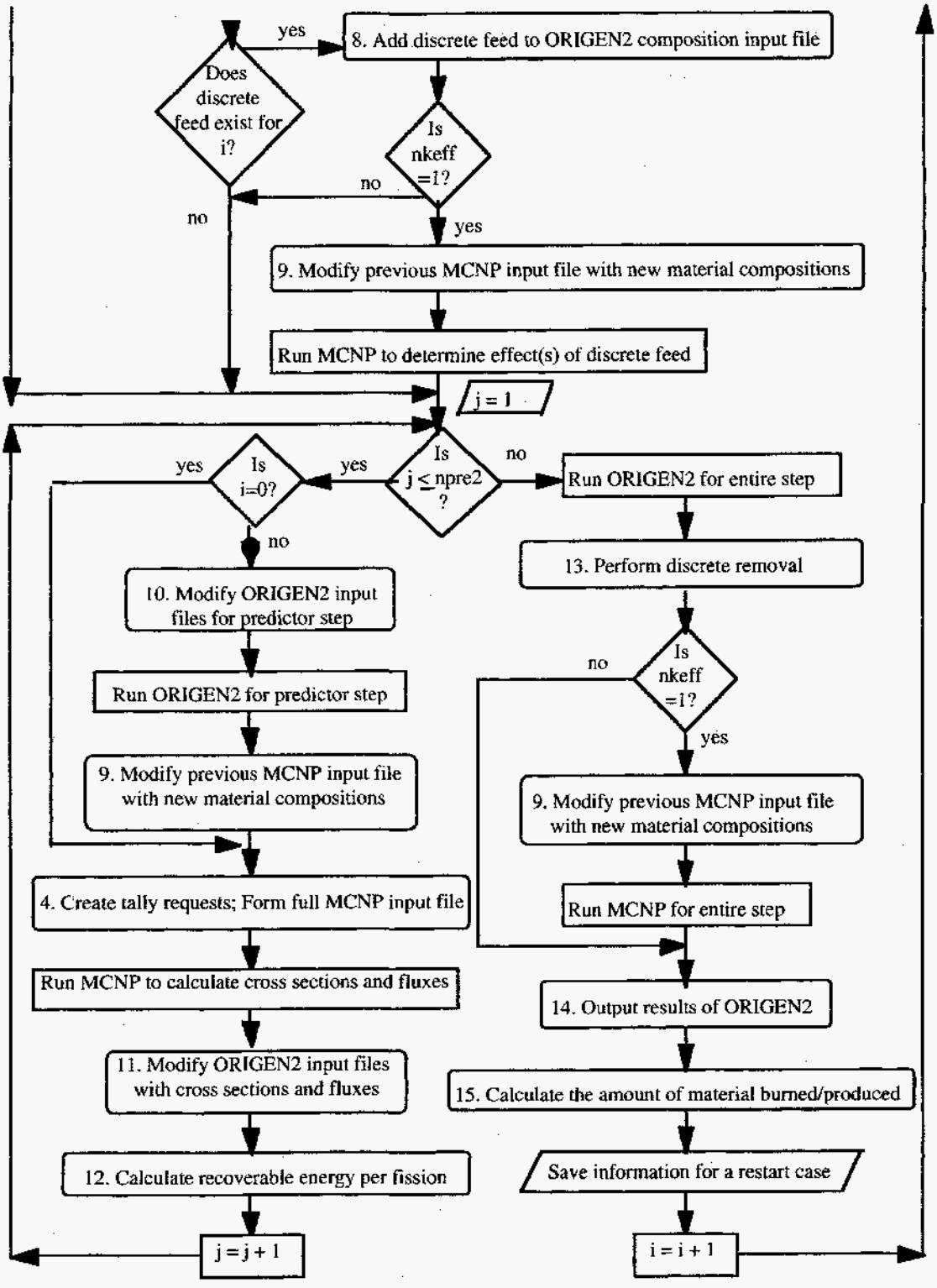


Figure 2. *Monteburns* Flow Chart

3.2 Calculations

The calculations performed by *monteburns* are divided into six different categories: recoverable energy per fission, flux normalization, reactor physics constants, effective multiplication factor, power, and importance fraction.

3.2.1 Recoverable Energy per Fission

The user has two options for calculating the recoverable energy produced per fission in a system. Either he/she can enter the desired Q-value (the average energy released by the entire system) into the *monteburns* input file, or the user can enter the Q-value for U-235 that he/she thinks is most representative for the nuclear system being evaluated (preceded by a negative sign in the input file), and the code calculates the average Q. In this case, the following equations are used by *monteburns* to calculate the recoverable energy produced per fission in each material (see Equation 8 for the Q-value of the entire system) according to the distribution of actinides in that material.

$$Q_{fis} = \left|Q_{U-235}\right| * Q_{rat} \tag{5}$$

where: Q_{fis} = total amount of recoverable energy produced per fission

Q_{U-235}= recoverable energy per fission for U-235 (input by user - recommended value is 200 MeV[16])

Q_{rat} = weighting factor to include recoverable fission energy for all actinides present (calculated by Equation 6)

$$Q_{rat} = \sum_{i=1}^{n} q_{rat}(i) * f_{rat}(i) \tag{6}$$

where: n = number of actinides in material (calculated by ORIGEN2)

$q_{rat}(i)$= ratio of recoverable energy per fission for isotope i divided by the recoverable energy per fission for U-235 (see Table 3)

$f_{rat}(i)$ = ratio of fissions resulting from isotope i to total number of fissions (calculated by Equation 7)

$$f_{rat}(i) = \frac{\sigma_f(i) * n(i)}{\sum_{i=1}^{n} (\sigma_f(i) * n(i))} \tag{7}$$

where: $\sigma_f(i)$ = spectrum-averaged one-group microscopic fission cross section of isotope i (calculated by MCNP)

$n(i)$ = number density of isotope i (calculated by ORIGEN2 in units of gram-atoms)

Next, the average energy produced per fission for the system as a whole is calculated.

$$Q_{ave} = \frac{\sum_{j=1}^{m} \left(Q_{fis}{}^{j} * \varphi_n^j * \Sigma_f{}^{j} * V^j \right)}{\sum_{j=1}^{m} \left(\varphi_n^j * \Sigma_f{}^{j} * V^j \right)}$$

where: Q_{ave} = average recoverable energy per fission for entire system (MeV)

$Q_{fis}{}^{j}$ = average recoverable energy per fission in material j (MeV) (calculated by Equation 5)

φ_n^j = neutron flux (n/cm^2-s) in region containing material j (calculated by MCNP)

$\Sigma_f{}^{j}$ = macroscopic fission cross section of material j (cm^{-1}) (= $\sum_{i=1}^{n} \sigma_f(i) * n(i)$ - obtained from ORIGEN2 files)

V^j = volume of all cells containing material j (cm^3) (calculated by MCNP or input by user)

m = number of materials being analyzed (input by user)

Table 3. Fraction of Recoverable Energy Per Fission for Certain Actinides Divided by the Recoverable Energy Per Fission for U-235

Isotope	Fraction[17] *	Isotope	Fraction
Th-227	0.90	Pu-240	1.04
Th-229	0.92	Pu-241	1.05
Th-232	0.96	Pu-242	1.06
Pa-231	0.95	Am-241	1.05
Pa-233	0.98	Am-242m	1.06
U-232	0.96	Am-243	1.07
U-233	0.99	Cm-242	1.06
U-234	0.98	Cm-243	1.07
U-235	1.00	Cm-244	1.08
U-236	1.00	Cm-245	1.09
U-237	1.01	Cm-246	1.10
U-238	1.02	Cm-248	1.12
Np-237	1.01	Cm-249	1.13
Np-238	1.02	Cf-251	1.15
Pu-238	1.02	Es-254	1.18
Pu-239	1.04		

* The fractions displayed here are an average of the fractions calculated for thermal and fast spectrums

3.2.2 Flux Tally Normalization

For each material j, the flux used in ORIGEN2 (see Equation 1) is calculated from the flux tallied by MCNP and is either normalized per MCNP fission neutron for a "kcode" source definition or per MCNP source neutron for "nps" source definition, both according to Equation 9.

$$\varphi = \varphi_n * C \qquad (9)$$

where: φ = true value of the flux (normalized to system power)

φ_n = flux tally normalized per fission or source neutron (from MCNP)

C = the neutron source term (calculated by Equation 10 or 11)

When an MCNP input file with a "kcode" (criticality) source definition is used, the flux is normalized per fission neutron, and the value of k_{eff} and its associated error are found in the MCNP output file. In this case, the value of C is given by Equation 10.

$$C = \frac{\upsilon * P * 10^6 W/MW}{(1.602 * 10^{-13} J/MeV) * k_{eff} * Q_{ave}} \tag{10}$$

where: υ = average number of neutrons produced per fission (calculated by Equation 12)

P = total power (MW) of system (input by user)

k_{eff} = effective multiplication factor (calculated by MCNP)

When the MCNP input file has a "nps" source definition, the flux is normalized per source neutron, and the value of C is instead:

$$C = \frac{src * P * 10^6 W/MW}{floss * (1.602 * 10^{-13} J/MeV) * Q_{ave}} \tag{11}$$

where: src = weight of source neutrons (approximately equal to one) (calculated by MCNP)

$floss$ = weight of neutrons lost to fission (calculated by MCNP)

The reason that the equation for the neutron source term has the variable k_{eff} (or *floss/src*, which represents the fraction of neutrons lost in fission in a "nps" source definition) in the denominator is that it modifies the value of the neutron flux of systems not modeled at critical. For a "kcode" problem, the flux calculated by MCNP is normalized per fission neutron, which assumes that the number of neutrons that fission in

the system modeled are representative of how many fission to produce the given steady-state power level (steady-state power is only produced at critical). However, if the system is subcritical, then the flux normalized per fission neutron is only a fraction (k_{eff}) of the flux produced at steady-state because only that fraction (k_{eff}) of neutrons in a steady-state system are represented. Dividing by k_{eff} increases the value of this flux appropriately. Similarly, the relative number of fission neutrons produced in a supercritical system are greater than those in a reactor at steady-state, so the flux must be reduced to accurately reflect power production. Additionally, a system designed to be subcritical (such as ATW) must rely upon source neutrons to remain at steady-state, and these neutrons are not included in the flux calculated by MCNP. Again, in both cases, dividing by k_{eff} produces the desired result.

The condition of a system not only influences the neutron flux in each region but also the energy spectrum. If the system modeled is subcritical but the actual system is critical, then the spectrum of the modeled system may not be representative of the actual one, cross sections may be inaccurate, and incorrect ratios of fission, capture, and leakage may be obtained. These three are competing processes that produce different nuclides (or none in the case of leakage) such that the resulting isotopic compositions of the system are affected by any misrepresentation of the spectrum. However, *monteburns* is not designed to account for such a spectrum shift in either direction. Instead, it only accounts for a linear change in the true flux as a function of $1/k_{eff}$. For a system designed to be subcritical (such as ATW), this effect is not as dominant because it does not have to be modeled exactly at critical throughout life to be representative of the actual system. In either case, it is recommended that user model a system such that k_{eff} at all time steps is as close to true values as possible so that the correct spectrum and results are obtained.

3.2.3 Reactor Physics Constants

For both types of source definitions, the value of υ (number of neutrons produced per fission) is calculated from results in the MCNP output file. For a "kcode" source definition, it is calculated using Equation 12.

$$\upsilon = k_{eff} * src/floss \tag{12}$$

where: src = weight of source neutrons (approximately equal to one) (calculated by MCNP)

$floss$ = weight of neutrons lost to fission (calculated by MCNP)

For a "nps" source definition, the value of υ is:

$$\upsilon = fsrc/floss \tag{13}$$

where: $fsrc$ = weight of source neutrons gained in fission (calculated by MCNP)

For either type of MCNP input file, the number of neutrons produced per neutron destroyed (η) in a material is:

$$\eta = \frac{(\upsilon\sigma_f + 2.0 * \sigma_{n2n})}{(\sigma_\gamma + \sigma_f + \sigma_{n2n})} \tag{14}$$

where: σ_f = fission cross section of material (calculated by MCNP)

σ_γ = (n,γ) cross section of material (calculated by MCNP)

σ_{n2n} = (n,2n) cross section of material[6] (calculated by MCNP)

---

[6] Additional cross sections for neutron interactions producing neutrons (i.e., (n,3n), (n,4n), etc.) are assumed to be negligible.

3.2.4 Effective Multiplication Factor

The value of the effective multiplication factor for a "nps" source definition must be calculated from the value of the net multiplication obtained from MCNP output:

$$k_{eff} = \frac{(fmult - 1)}{(fmult - 1/\upsilon)} \tag{15}$$

where: *fmult* = net multiplication in the system (calculated by MCNP)

The relative error (σ) associated with k_{eff} is then:

$$\sigma = \left\{ \frac{(fmult*(1+err) - 1)}{(fmult*(1+err) - 1/\upsilon)} - k_{eff} \right\} / k_{eff} \tag{16}$$

where: *err* = relative error associated with net multiplication in system (calculated by MCNP)

3.2.5 Power

Finally, the power produced by each material is:

$$P^j = \frac{\left(Q_{ave} * \varphi^j * \Sigma_f{}^j * V^j * 1.60219 * 10^{-13} J / MeV\right)}{10^6 W / MW} \tag{17}$$

where: P^j = power produced by material j (MW)

φ^j = neutron flux (n/cm^2-s) in region containing material j (calculated by Equation 9)

3.2.6 Importance Fraction

A key factor in balancing accuracy with execution time in *monteburns* is determining the number of isotopes for which spectrum-averaged one-group cross sections are calculated in MCNP. It is important for isotopes to be included in MCNP

for two primary reasons: they may significantly affect the system flux spectrum and reactivity, and/or an MCNP modified spectrum-averaged one-group cross section produces more accurate transmutation and fission rates in ORIGEN2. For some isotopes it may be important to modify this cross section, while for others, the default ORIGEN2 value may be used with little effect on the accuracy of the solution. Thus, it is inefficient to calculate a spectrum-averaged one-group cross section for every isotope included in the associated MCNP libraries because it increases execution time, although this can be done if desired. Isotopes are deemed "important" in two ways. The first way is to explicitly list an isotope in the *monteburns* input file (i.e., designate it as an "automatic" isotope); this insures that spectrum-averaged one-group cross sections are calculated for this isotope during each burn step (and that this isotope is included in the primary *monteburns* output). The other way in which an isotope is deemed "important" is based on a user input variable called the importance fraction.

If an isotope contributes a fraction to the system neutron absorption, fission, mass, or atom density higher than the importance fraction, then this isotope is deemed "important," and a spectrum-averaged one-group cross section is calculated in MCNP and modified in ORIGEN2. If any of the values calculated by Equations 18-21 (fraction of absorption, fraction of fission, weight fraction, and atom fraction respectively) are greater than the value of the importance fraction assigned by the user, then the isotope is considered "important" and is included in all transfers between ORIGEN2 and MCNP for the remainder of the run.

$$f(\sigma_a)_i = \frac{gad_i * \sigma_{ai}}{\sum_{i=1}^{n} (gad_i * \sigma_{ai})} \tag{18}$$

$$f(\sigma_f)_i = \frac{gad_i * \sigma_{fi}}{\sum_{i=1}^{n} (gad_i * \sigma_f)} \tag{19}$$

$$w_{fi} = \frac{gad_i * A_i}{\sum_{i=1}^{n} (gad_i * A_i)} \tag{20}$$

$$a_{fi} = \frac{gad_i}{\sum_{i=1}^{n} gad_i} \tag{21}$$

where: n = total number of isotopes in system (input by user)

$f(\sigma_a)_i$ = fraction of absorption that isotope i contributes to system

gad_i = amount of isotope in system (gram-atoms)
(calculated by ORIGEN2)

σ_{ai} = microscopic absorption cross section of isotope i
(obtained from ORIGEN2 library or calculated by MCNP)

$f(\sigma_f)_i$ = fraction of fission that isotope i contributes to system

σ_{fi} = microscopic fission cross section of isotope i
(obtained from ORIGEN2 library or calculated by MCNP)

w_{fi} = weight fraction of isotope i in system

A_i = atomic weight (grams) of isotope i (calculated by *monteburns*)

a_{fi} = atom fraction of isotope i in system

In this document (and within *monteburns*), the word "absorption" solely refers to capture interactions (primarily (n,γ)) and excludes the probability of fission. Nonetheless, both types of interactions influence the value of k_{eff} and what occurs to the neutrons in a system (i.e., if a neutron is absorbed in a material, its "life" ends, whereas if that absorption leads to fission, it produces even more neutrons as a result). If an isotope significantly contributes to either one or both of these areas, it is included in further MCNP calculations.

Not all isotopes produced from irradiation interactions are included in the initial ORIGEN2 cross section libraries and are thus not deemed "important" by their absorption or fission contribution because their cross sections are effectively zero. If such an isotope comprises a significant portion of the material (either by weight or atom density), then it should also be included in MCNP because it could significantly contribute to interactions in the system. Thus, if the weight and/or atom fraction of an isotope in ORIGEN2 is greater than the importance fraction, then the isotope is also passed back to MCNP. Additionally, even if an isotope does not have an absorption or fission fraction greater than the importance fraction but still exists in a material in significant amounts, it may still contribute to scatter interactions in the system. By allowing the atom and weight fractions to be included in "importance" checks for an isotope, such a potential scatterer can be included.

3.3 User Input

The user must generate two to four different input files before executing *monteburns*. The two required input files are the MCNP input file (designated here by *mbfile* but can be any name up to 8 characters), and a general *monteburns* input file (this must have the same prefix "mbfile" with an extension of ".inp" for a name of *mbfile.inp*). For many complex burnup scenarios, the user must also generate a feed input file (with a name of *mbfile.feed*), which contains detailed instructions for *monteburns* at each time step (i.e., time interval, power, material feed/removal). The only case in which a feed input file is not required is for a constant power burn with no material feed or removal. Finally, *monteburns* uses one other input file, *mbxs.inp,* which contains a list of default MCNP cross section identifiers for isotopes that may be produced in the irradiation process and are not initially specified by the user.

3.3.1 MCNP Input File

The MCNP input file represents the system being analyzed, including the geometry and compositions of materials. There is no required format of this input file in *monteburns*, except that material numbers must not be greater than 100 and user tally cards cannot have numbers greater than 100 (this is to keep *monteburns* tallies from interfering with user input). This file must run in MCNP (for example, complete enough active cycles to produce a "final result" for k_{eff} in a "kcode" problem) before it can work in *monteburns*.

3.3.2 Monteburns Input File

The following pages list input parameters required for *monteburns* that must be provided by the user in the *monteburns* input file. These input parameters are read in free format, but they must be in the order listed below (for more information, see the *Monteburns* User's Manual[18]). In addition, sensitivity analyses were performed for several input parameters to see how their values affected results. The outcome of these analyses is located in Section 4.2.1.

- **Number of MCNP Materials** - this indicates the number of materials the user wants to irradiate from the MCNP input file (i.e., transfer back and forth between MCNP and ORIGEN2).
- **MCNP Material Number(s)** - the identification number of the material(s) in the MCNP input file for which a burnup analysis is desired (the average flux for all cells and parts of a repeated structure or lattice with this material are obtained). Note: the number of entries here must equal the number of MCNP materials entered above.
- **Material Volume(s)** - the sum of the volume (cm^3) of all cells in the MCNP input file for each material number(s) listed above (again, the number of entries must equal the number of MCNP materials). If the user enters a value of 0.0 for one or more of these, then the volume calculated by MCNP is used (if it exists). However, often the

geometry is too complex for MCNP to calculate the volume, in which case, unless the user has input a non-zero volume for that material number, an error message appears, *monteburns* terminates, and it must be rerun with non-zero values. Additionally, in most cases of repeated structures, MCNP calculates the volume of cells containing a given material incorrectly. For each of these cases (and for any other instances the user desires), the user must enter the sum of the volumes of cells containing each material being analyzed.

- **Total Power of System** - the power (MW) generated by the entire system represented in the MCNP model (note: this is not necessarily the same as the power generated solely by the materials burned in *monteburns*). This value, along with the recoverable energy per fission, is used to normalize the flux from MCNP in each burned region for ORIGEN2. This flux is then converted to fission power and output. Additionally, the user can enter the fraction of this power to be used during each outer burn step (if power is not constant over the entire burn) in the feed input file. By entering a power fraction of zero for a step, then it effectively becomes a decay-only step, which is useful for analyzing cooling periods of systems. Note: the value of fission power output is subject to statistical errors and may not be exactly the same as the power input. Increased statistics in MCNP may minimize this problem, but nonetheless, the user should check the value of power output to ensure that it is close to the amount of power desired.
- **Recoverable energy per fission** - this value represents the average recoverable energy per fission (Q) in MeV in the aforementioned MCNP model. If the user does not know the exact amount of energy generated by a combination of several isotopes, then he/she can enter the recoverable energy per fission for U-235 in that system (see Equations 5-8). WARNING: the fissile isotopics used for the calculation of Q are based only on the materials burned by *monteburns*. If the fissile isotopics of the entire system are significantly different from the fissile isotopics of the materials

burned, then the average value of Q may be in error, thus the flux normalization may be incorrect (although in most cases this should be a relatively small effect).

- **Total number of days burned** - this number represents the length of time for which a material is irradiated in ORIGEN2 (or the decay time if power equals zero). If the user provides a feed input file, then the irradiation lengths (in days) for each outer burn step (described below) must be provided in this file. Otherwise, the total irradiation time (in days) is entered in the *monteburns* input file.
- **Number of outer burn steps** - this number indicates how many outer burn steps are desired. If a feed input file exists, then this must equal the number of steps described in the feed input file. If a feed input file does not exist, then the length of the irradiation period for each outer burn step equals the total days burned divided by the number of outer burn steps. Each of these steps represents a time period for which a burnup calculation is performed and representative cross sections are obtained (the burn step then uses spectrum-averaged one-group cross sections calculated at a predictor step halfway through that step). Each outer step can also indicate the addition and/or removal of a material.
- **Number of internal burn steps** - this is the number of additional times into which the irradiation period is divided for ORIGEN2 calculations. As mentioned in Section 2.2, the results obtained from ORIGEN2 (and as a result, *monteburns*) may be more accurate if long irradiation periods are broken up into smaller lengths of time, especially at the beginning of a system's life. This is because the Bateman equations and/or the Gauss-Seidel iterative technique are used to solve for compositions of materials when the half-life of an isotope is less than 10% of the irradiation interval.[6] Additionally, the physics and composition of materials in the system may change significantly with time. Thus, the user can specify that the outer burn steps be divided into even smaller time segments for use in ORIGEN2. In addition, there is

virtually no penalty on execution time by using smaller time steps in ORIGEN2 because most of the execution time lies with MCNP.

- **Number of predictor steps** - this is another variable affecting the accuracy of the results. As the isotopic composition of a material changes during an irradiation step (both due to burnup and potential variances in continuous feed from beginning to end), the cross sections may change as well. To obtain the most accurate results, spectrum-averaged one-group cross sections for a burn step should represent an average over the time interval. In a *monteburns* calculation, ORIGEN2 is run halfway through each outer burn step, and the resulting isotopics are used in MCNP to calculate spectrum-averaged one-group cross sections and fluxes for that step. Then a complete ORIGEN2 run is performed with the new values to determine final compositions. This assumes that the isotopics of the system at the midpoint are a reasonable approximation of the isotopics over the entire burn step and that cross sections are representative of the step (actually it is only important that the neutron flux energy spectrum is representative of the entire burn step). The user must be aware of this assumption, and consequently, ensure that burn intervals are not too long.

  If the initial cross sections for a step are not accurate, then the ORIGEN2 compositions halfway through the step may not be a good representation of the burn step. Thus, it is often beneficial to perform a "predictor" step (derived from a basic form of the predictor-corrector method[8]) to calculate cross sections more than once at the midpoint of a burn step and to compare the neutron energy spectrum and isotopic compositions halfway through the step (these values are printed in the output files) to make sure that the final cross sections are representative of the system at that step. The number of times for which cross sections are calculated halfway through each step is the number of predictor steps. Executing multiple predictor steps increases the accuracy of the burnup calculation because the

spectrum-averaged one-group cross sections used to perform the predictor step approach the ones calculated by the predictor step (i.e., they converge). In addition, *monteburns* automatically adds a predictor step for the initial burn step because the actual spectrum-averaged one-group cross sections for a system may be different than those supplied in the chosen default ORIGEN2 library. For all subsequent burn steps, *monteburns* uses the modified spectrum-averaged one-group cross section library from the previous burn step, thus an extra predictor step is not required.

- **Step to restart after** - a user can use this parameter to restart a run that ended unexpectedly, or to branch off from a previous *monteburns* run with different input variables (for example, if k_{eff} drops too low during the n^{th} burn step, the user can change the feed rate for the n^{th} step and restart from the previous step). The "restart step" indicates the outer burn step after which *monteburns* should start, using all previously created input files and results for the outer burn steps up to that point. To use this variable effectively, all input files that were created by *monteburns* during the previous run must remain in the directory in which *monteburns* is running (most of these appear in the *tmpfile* subdirectory of the main directory). If a restart run is not being performed, then the "restart step" value should be zero. This value gets modified during each step to reflect the value of the current step.
- **Number of ORIGEN2 library** - this number represents the number of the ORIGEN2 library from which initial one-group cross sections are obtained (these values are then modified to be system-dependent as calculated by MCNP tallies after the first step for "important" isotopes). The ORIGEN2 manual[2] contains a list of over forty different cross section libraries (with two-digit identifiers) from which the user can choose for different types of systems. The value of this two-digit identifier must be entered by the user.
- **ORIGEN2 library location** - this line of input must contain the location of the ORIGEN2 libraries (both decay and cross section ones) in the user's file space or in

the directory of another user on the system that has the library files. This way, only one user on a UNIX operating system needs to have a copy of the libraries.

- **Importance Fraction** - this value represents the lower limit (tolerance) for the importance of one isotope relative to the rest of the system based on results obtained from ORIGEN2 and MCNP. If an isotope contributes a large enough fraction (i.e., greater than the importance fraction) to absorption or fission interactions, mass, or atom density (see Section 3.2.6 for more information), then the isotope is considered "important." Flux and one-group spectrum-averaged cross sections tallies are then performed in MCNP for this isotope. If the importance fraction is zero, then all activation, fission products, and actinides generated in ORIGEN2 are tallied (except those for which no MCNP cross section exists - see Section 3.3.4 for more information). If the importance fraction is one, then no isotopes are deemed "important" except those specified as "automatic" in the input. Additionally, it is advised that the initial ORIGEN2 library be somewhat representative of the system, or "important" isotopes may not be properly identified. The only way to absolutely avoid this problem is to track every isotope or to generate a problem specific library with a previous run of *monteburns* that replaces the original default ORIGEN2 library.

  The user must also decide how to deal with fission products. If the user enters the importance fraction as a positive value, then only those fission products deemed "important" are included in MCNP. However, since MCNP cross sections for many fission products do not exist, *monteburns* contains the option to lump all fission products together as one sum (except for those fission products, if any, designated as "automatic" in the *monteburns* input file) by using a negative value here. These lumped fission products are then given one of two general fission product cross sections in MCNP - the average fission product from Uranium-235 and the average fission product from Plutonium-239 (these have the identifiers 45117.90c and

46119.90c respectively[11]). The fraction of the total fission product mass separated into each category is determined by comparing the number of fissions that result from isotopes with an atomic number less than or equal to that of uranium (92) to those that occur in other transuranic actinides with an atomic number greater than 92.

- **Intermediate flag** - this flag indicates whether intermediate k_{eff} calculations are performed. Normally, MCNP is only run once per predictor step, and these runs occur halfway through each outer burn step (i.e., halfway through each irradiation period). However, it is often desired to obtain a value of k_{eff} at the beginning and/or end of each burn step. When the value of this parameter is one, these additional MCNP calculations are performed. Neither cross sections nor fluxes are recalculated by MCNP for these runs, so ORIGEN2 results are not influenced. The only purpose "intermediate" MCNP calculations have is to provide the value of k_{eff} at more than one point during each outer burn step to see how the system changes. When a discrete feed addition (see Section 3.3.3) occurs, three MCNP runs are performed for the step (at the beginning, middle, and end); otherwise two MCNP runs are performed (at the middle and end) because the beginning value of k_{eff} equals the ending value of k_{eff} from the previous step. If the value of this parameter is zero, then only one MCNP run is performed for each outer burn step (in the middle) regardless if discrete feed occurs.
- **Number and list of automatic tally isotopes for each material** - this integer represents the number of isotopes/elements for which the user wants tallies to be performed in MCNP and results written to *monteburns* output files (i.e., automatic "important" isotopes). The user must then enter the MCNP identification number for each of these isotopes/elements (these can indicate library preference and/or temperature dependence). It also allows the user to use a cross section not specified in the default cross section file discussed in Section 3.3.4, *mbxs.inp* (i.e., the cross section identifier listed here has precedence over the one in *mbxs.inp*).

3.3.3 Feed Input File

The purpose of a feed input file in *monteburns* is to list the lengths of each time step, to vary the fraction of power generated by the system during each time steps, to shuffle materials from one region to another, and/or to specify amounts of materials to add to or remove from the system during each outer burn step. The user can also specify continuous or discrete (all at one time) feed (addition of isotopes) and/or removal (of specified elements) for each material at each time step in this file. First, for each outer burn step and (excluding the first two items) material, the user enters the following parameters:

- length of the irradiation (in days),
- fraction of power produced relative to the total power entered in the *monteburns* input file,
- region in which each material is located,
- feed group (defined below),
- feed rate(s) (both beginning and ending rates for continuous and a flag and a rate for discrete),
- removal group (positive for continuous feed, negative for discrete), and
- removal fraction (the fraction of each element removed (for example, a fractional removal of 0.9 means that 90% of the removal group is removed and 10% remains)).

The next part of the feed input file allows the user to enter information about the feed group(s). This includes:

- the number of feed groups,

then, for each feed group,

- the number of isotopes in that group, and

- a list of those isotopes (atomic number followed by atomic mass number (for example, 92235 for U-235)).

Continuous feed occurs at several points throughout the irradiation process, the amount of feed being interpolated from the beginning and ending rates, and discrete feed occurs all at the beginning.

The final part of the feed input file consists of information about the removal group(s), including:

- the number of removal groups,

then, for each removal group,

- the number of ranges of elements to be removed,
- the range(s) of elements (for example, 28 to 68 means that all elements between nickel and erbium are removed (which represents a majority of fission products), the two ranges 28 to 42 and 44 to 68 mean that all fission products in this same range except technetium (Z=43) are removed, and the range 43 to 43 indicates that only the element technetium is being removed).

For continuous removal (a removal group number greater than 0), the appropriate elements are removed both after the halfway predictor step and at the end of the burn (simulating continuous removal), whereas for discrete removal (a removal group number less than 0), the elements are removed only at the end of the burn step.

3.3.4 Identifier Input File

The identifiers used to recognize isotopes in MCNP are different than those in ORIGEN2. Thus, *monteburns* is designed to determine which identifiers to use for each code. In ORIGEN2, the identifier is simply the atomic number followed by the atomic mass number and a "0" for most isotopes (metastable isotopes are followed by a "1"). MCNP not only requires the atomic number and atomic mass number but also a cross

section identifier. A file containing a list of default MCNP identifiers for all isotopes used or potentially created by decay or irradiation processes must be present in the directory in which the user is running (Note: cross section libraries for many fission products may not exist and obviously cannot be listed here). This file is named *mbxs.inp* and can either be provided by the user or obtained with the source code and modified by the user as necessary. For any isotopes deemed "important" by *monteburns* but do not have a cross section identifier in this file, *monteburns* gives a warning that the cross section is not found, continues to use the default ORIGEN2 cross section, and does not transfer the material to MCNP. The identifiers in this file can either be cross section libraries provided by MCNP, or they can be ones generated by the user with ENDF libraries and/or the code NJOY,[7] or ones from other sources. In fact, the user is encouraged to use a code such as NJOY to generate temperature-dependent cross section libraries, which can then be used by MCNP/*monteburns* to process temperature-dependent data. In addition, *mbxs.inp* must include the general fission product identifiers 45117.90c and 46119.90c for MCNP if the lump sum of fission products option is used (as discussed in Section 3.2.6 and 3.3.3).

There are a number of elements in MCNP for which "natural" cross sections exist. However, ORIGEN2 does not recognize natural elements, so *monteburns* contains data to separate natural elements into individual isotopes. If a natural cross section exists in the MCNP input file, *monteburns* separates this element into its isotopic components, and then ORIGEN2 burns these isotopes individually (with the default ORIGEN2 library cross sections). After the ORIGEN2 burn, *monteburns* then lumps them back into the element's natural isotopics for use in MCNP. Although this may not be completely accurate because the initial ORIGEN2 cross sections are not modified by MCNP (i.e.,

---

[7] Versions of NJOY are available at the Radiation Safety Information Computational Center (RSICC) as codes PSR-171 and PSR-355.

they are not fully representative of the material in the system), it is dictated by the lack of MCNP cross sections for many individual isotopes.

3.4 Output

Two large, primary output files are produced by *monteburns*. These output files consist of the name of the MCNP input file created by the user followed either by the extension ".mbout" or ".mbchk." For each of the output groups listed below (except the first two, which contain system, not material dependent parameters), results appear for each *monteburns* material/region being analyzed. Note: this is not necessarily the same as the initial MCNP material number assigned to each region due to shuffling between regions. The user must keep track of each MCNP material individually through the various regions when shuffling occurs.

The first output file, *mbfile.mbout*, contains the results displayed below for each outer burn step:

- ***Monteburns* MCNP k_{eff} Versus Time** - a list of the cumulative time (in days) over which irradiation has occurred as well as the effective multiplication factor (k_{eff}), associated relative error, υ (see Equations 12 or 13), average recoverable energy per fission calculated by *monteburns* (see Equations 5-8), and η for the system (see Equations 8 and 14 respectively).
- ***Monteburns* MCNP k_{eff} at Beginning of Step** - a list of the cumulative time of irradiation (in days) that has occurred before each step begins as well as the effective multiplication factor, relative error, and υ at the beginning of each outer burn step (after discrete feed occurs). This data is only included in the output if discrete feed is used and intermediate k_{eff} calculations are requested.

For each material and outer burn step, the following parameters are output:

- ***Monteburns* Transport History** - the recoverable energy per fission (see Equation 5), neutron flux (see Equation 9), macroscopic fission cross section (Σ_f), power generation, burnup (in gigawatt-days per metric ton heavy metal (i.e., actinides) (GWd/MTHM)), capture - (n,γ), fission - (n,f), and (n,2n) cross sections, fission-to-capture ratio, and η (see Equation 14) for both the material as a whole and the actinides only.
- ***Monteburns* Flux Spectrum** - the percent of neutrons with energies in each of the following ranges: 0 to 0.1 eV, 0.1 to 1 eV, 1 to 100 eV, 100 eV to 100 keV, 100 keV to 1 MeV, and 1 MeV to 20 MeV. To obtain a more detailed spectrum, the user must enter his/her own tallies into the MCNP input file or modify *monteburns* to provide the values desired.

The following results are provided for each "automatic" isotope in each material for each outer burn step:

- ***Monteburns* One-Group (n,γ) Cross Sections** - the value of the microscopic capture cross section (σ_c). This capture cross section is assumed to be equal to the (n,γ) cross section for the isotope, which is its primary constituent. Other reactions, such as (n,p), (n,d), (n,t), etc. may contribute to the total capture cross section, but not in significant amounts.
- ***Monteburns* One-Group Fission Cross Sections** - the value of the microscopic fission cross section (σ_f).
- ***Monteburns* Fission-to-Capture Ratio** - the ratio of the microscopic fission cross section to the microscopic capture (n,γ) cross section (σ_f / σ_c).
- ***Monteburns* Grams of Material at Beginning of Steps** - this represents the amount of material (in grams) that exists in the system at the beginning of each step.

- ***Monteburns* Grams of Material at End of Steps** - the amount of material (in grams) at the end of each step.
- ***Monteburns* Grams of Feed** - the amount of material (in grams) added to the system.
- ***Monteburns* Grams Produced (or Destroyed)** - the amount of material (in grams) produced (or destroyed if the output is negative) during irradiation. The interpretation of this data may depend on feed, removal, and/or material shuffling.
- **Summary of Inventory/Feed/Production** - the total amount of material in the system at the beginning and end of *monteburns* (not of each step), the amount added through feed, and the net change. The interpretation of this data may also depend on feed, removal, and/or material shuffling.
- **Feed Rate** - the average continuous feed rate (in grams per day).
- **Production/Destruction Rate** - the rate of change (in grams per day) of material produced to that destroyed during irradiation. The interpretation of this data may depend on feed, removal, or material shuffling.
- **Feed Input File** - if it exists, this file is included at the end of this output file so that the user can determine what feed parameters he/she used to produce the results presented in this output file.

In the second output file, *mbfile.mbchk*, many intermediate results from the execution of *monteburns* are listed. In this output file, the following results are reported for each *monteburns* material analyzed for each predictor step:

- ***Monteburns* Spectrum for Each Predictor** - the percent of neutrons with energies in each of the following ranges: 0 to 0.1 eV, 0.1 to 1 eV, 1 to 100 eV, 100 eV to 100 keV, 100 keV to 1 MeV, and 1 MeV to 20 MeV. This can be used to determine if smaller time intervals or more predictor steps need to be run.

- ***Monteburns* Grams at Midpoint** - the amount of each isotope (in grams) present halfway through the irradiation for both the predictor and the actual steps. The grams of each automatic "important" isotope present halfway through each predictor step are listed first for each outer burn step followed by the composition of these isotopes halfway through the actual step. This way the user can determine if the predictor step(s) provided enough accuracy or if more predictor steps (or smaller time intervals) are needed. If the two values for any isotope are significantly different, then *monteburns* should be rerun using more predictor steps or outer burn steps to obtain more representative cross sections.
- **Importance Fraction of Isotopes Sent From ORIGEN2 to MCNP** - the isotopes deemed "important," both automatically and through the importance fraction. This list contains the total mass of the isotope in the specified region and the contribution of each isotope in the following four categories: absorption, fission, mass fraction, and atom fraction. For example, if the fission column for an isotope reads 0.1, then 10% of the fissions resulted from this isotope. This file also includes a warning message if an isotope deemed "important" by *monteburns* or "automatic" by the user is not found in the MCNP cross section library used by *monteburns*.

4.0 BENCHMARKING/STATISTICS

One of the most important aspects of developing a new computer code *is* benchmarking it against existing experimental data and/or published calculations from other codes. The linkage code *monteburns* is no exception. To show that it is capable of performing burnup calculations well, a variety of test cases were run. Statistical analyses were also performed for selected input parameters and various system models to determine how they affect the outcome. Results from the benchmarking and the statistical analyses are presented in this section.

4.1 Benchmarking

The benchmarking process for *monteburns* used five different test cases, representing a variety of burnup scenarios. These test cases show the versatility of *monteburns* in performing all types of burnup calculations. First, changes in the concentrations of uranium and plutonium isotopes were calculated as a function of burnup, and then both a pin in a simple cell geometry and a full reactor assembly were analyzed. The first three test cases examined a PWR system and low-enriched uranium (LEU) fuel, the fourth involved a Boiling Water Reactor (BWR) system, and the fifth used mixed-oxide (MOX) fuel. The broad range of these cases is useful in showing the validity of *monteburns* in handling a variety of parameters. All cases were modeled using temperature-dependent cross sections derived from the ENDF/B-V data set and processed by NJOY.[14] Brief descriptions of these five test cases are:

1. Uranium and Plutonium Isotopic Concentrations as a Function of Burnup
2. Composition of Isotopes in a Fuel Pin at Fixed Burnups
3. Concentrations of Isotopes in a PWR Lattice at Fixed Burnups
4. Power Distribution of Pins Within a Small BWR Lattice
5. Activity of MOX-Based Spent Fuel After Removal from a Reactor

4.1.1 Isotopic Concentration

The first test case involved tracking the weight percents of several uranium (U) and plutonium (Pu) isotopes as well as fission products (FPs) in a typical PWR system as a function of burnup.

4.1.1.1 Description

A number of textbooks and other sources have published this information, and one representative figure[7] was compared to the results obtained by *monteburns* for a standard Westinghouse PWR assembly.[19] The *monteburns* output is shown in Figure 3a, and the isotopic concentrations calculated by basic burnup equations in Ref. 7 appear in Figure 3b.

4.1.1.2 Results

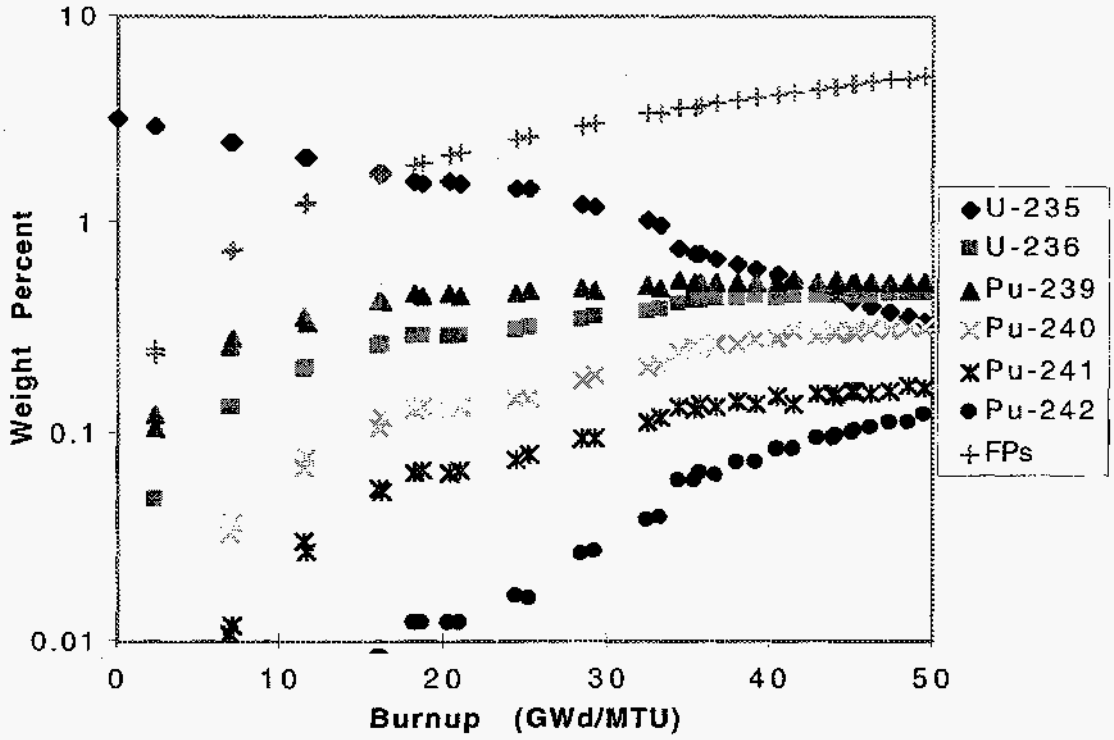


Figure 3a. Calculated Isotopic Distribution as a Function of Burnup as Predicted by *Monteburns*

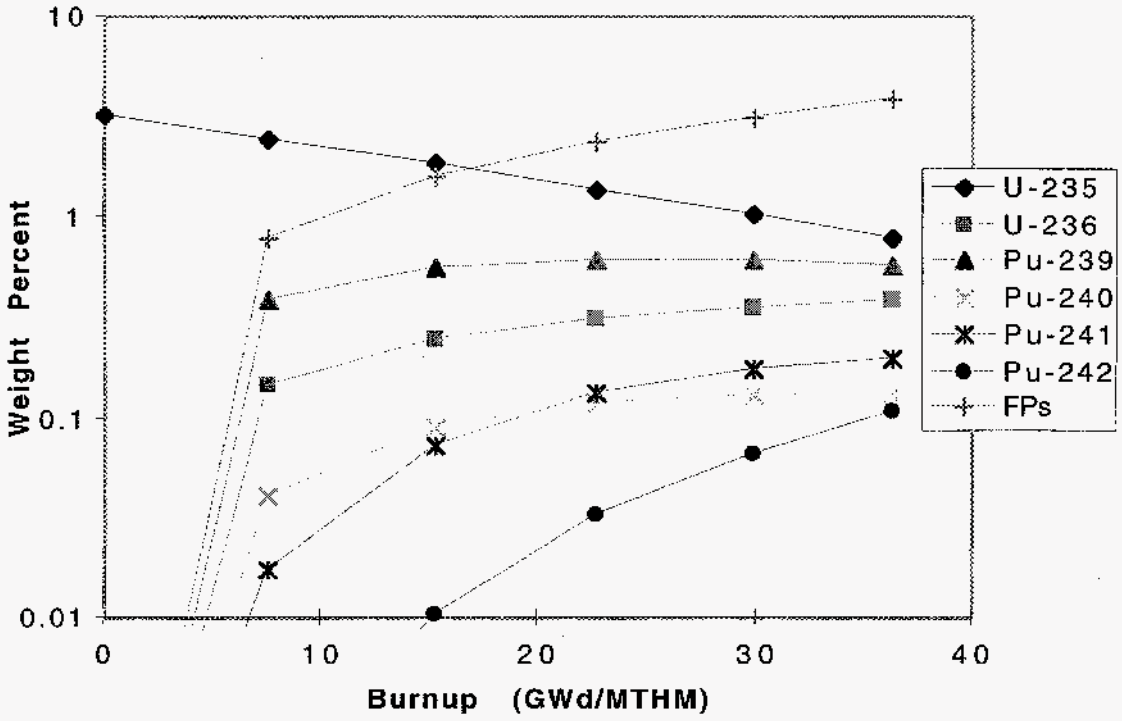


Figure 3b. Published[7] Isotopic Distribution as a Function of Burnup

The differences seen for actinides are discussed in terms of two categories: resonance self-shielding, and cross sections. Then variances in fission product concentrations are discussed.

4.1.1.3 Resonance Self-Shielding

Figures 3a and 3b display fairly similar results, with the exception of the isotopes Pu-240, Pu-241, and Pu-242. This variance was expected because, as the text in Ref. 7 states, the burnup equations that generated Figure 3b used one-group effective thermal cross sections for a PWR and did not account for resonance absorption, self-shielding, or the change in cross sections with burnup as *monteburns* does. When a system is initially started, it has a thermal spectrum, which means that a majority of neutrons in the system are at relatively low energies and are more likely to be absorbed than if they were at higher energies (the absorption cross section is higher at thermal energies because of 1/v

effects[16] (i.e., cross sections are indirectly proportional to neutron energy)). As burnup in a system increases, the number of isotopes built into the system also increases. The creation by fission and absorption of additional isotopes adds new resonance energies to the system in the resonance region (approximately 0.1 eV to 3 keV).[13] Neutrons created by fission typically have energies greater than 50 keV, and as they slow down (assuming enough moderator exists), they can be absorbed in resonances. If many of these neutrons are absorbed in the first (highest energy) resonance, then the neutron flux that would otherwise go to resonances at lower energies (and consequently, the total amount of resonance absorption) would decrease. The flux around this resonance is also depressed because many neutrons at that energy are absorbed, decreasing the flux seen by the fuel.[4] Thus, resonance self-shielding (as this process is called) can significantly decrease the neutron flux in regions of multiple, closely-spaced resonances.

The one-group cross section for an isotope is calculated by weighting the absorption cross section at each energy by the neutron flux at that energy, and having low fluxes at energy(ies) with large absorption cross sections (i.e. resonances) decreases the overall one-group absorption cross section of many actinides. The energy spectrum then either becomes more soft or more hard, depending on the ratio of neutrons that exist in the thermal energy region (below the resonances) to those in the fast region (above the resonances).

Additionally, as plutonium is built into the system during irradiation, the energy spectrum of neutrons somewhat hardens because the absorption cross sections of several plutonium isotopes are larger than those of uranium ones, and a large thermal resonance exists for Pu-239 and Pu-241 at an energy lower than that of the resonances of U-235 and U-238 (about 0.1 eV compared to around 5 eV). Thus, the neutron flux in both the thermal and the resonance regions decreases with burnup because as additional plutonium is built into the system, more absorption occurs (due to a larger absorption cross section), and the spectrum slightly shifts to higher energies.

The effects of resonance self-shielding are especially significant for Pu-240, which has a large absorption cross section at resonance energies, but as burnup increases, the cross section of this isotope significantly decreases (363 to 92 barns) due to resonance self-shielding. Figure 3c shows the variance in the compositions of Pu-240, Pu-241, and Pu-242 between the concentrations calculated by the equations in the reference (eq) and those calculated by *monteburns* (mb). The amount of Pu-240 calculated by *monteburns* was greater than that calculated by the equations in Ref. 7 because less of it was depleted through absorption interactions (i.e., σ_a was lower). It follows that the concentrations of Pu-241 and Pu-242 were smaller in *monteburns* than the referenced equations because less of them were built up from neutron absorption in Pu-240. However, all concentrations seen in Figure 3a do match those obtained using another code, CELL.[7]

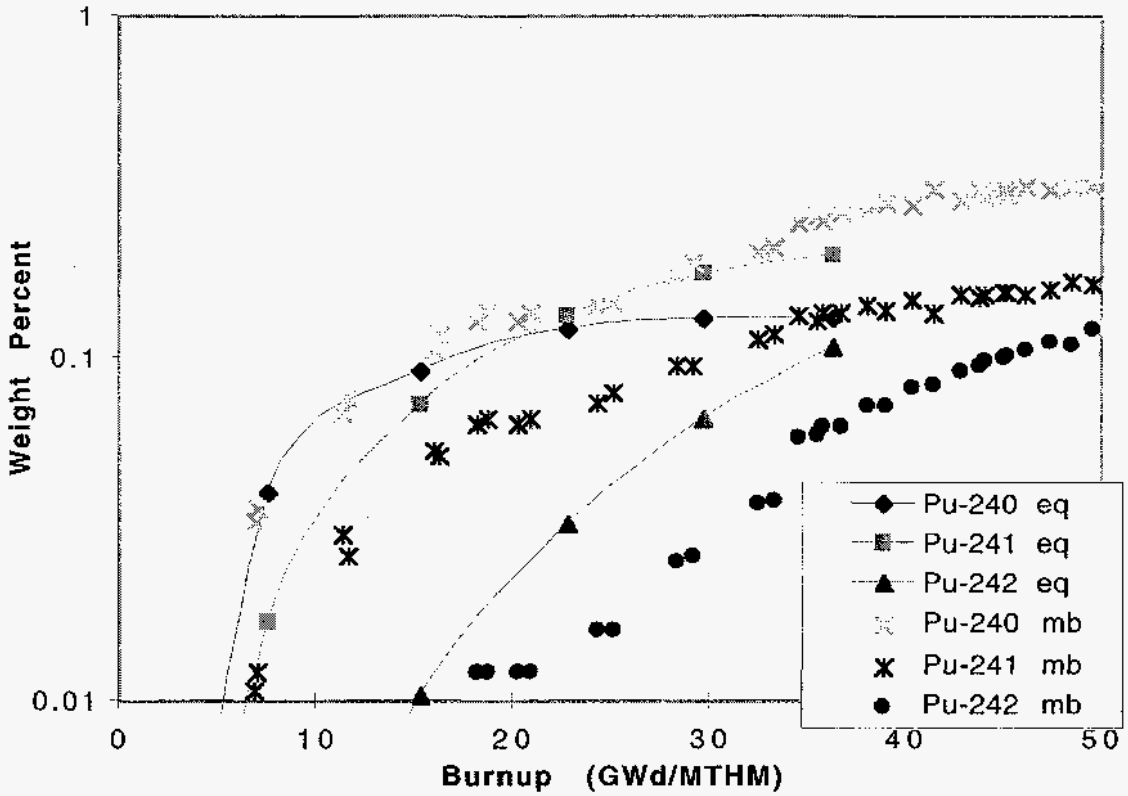


Figure 3c. Differences in Higher Isotopes of Plutonium
eq means the equations in the reference and mb stands for *monteburns*

4.1.1.4 Cross Sections

Another parameter compared in this analysis was the fission-to-capture ratios of the uranium and plutonium isotopes analyzed (see Table 4a). The fission-to-capture ratio in Ref. 7 for U-235 was smaller than the ratio calculated by *monteburns* at the various burn steps. This means that more fissions occurred per U-235 atom in *monteburns* than in the reference, causing more to be burned (as can be seen by comparing Figures 3a and 3b). The fission-to-capture ratio of U-238, however, decreased slightly with burnup, which means that the rate of capture slightly increased relative to the rate of fission, producing a few more plutonium atoms in *monteburns*. In addition, the fission-to-capture ratios of plutonium isotopes increased slightly as a function of burnup in response to the decrease of the absorption cross sections due to resonance self-shielding. For Pu-240, even though its fission-to-capture ratio increased, its probability of fission was so small that it was still not depleted as rapidly as when constant cross sections that did not account for resonance self-shielding were used and more transmutation occurred.[7]

Table 4a. Comparison of the Change in the Fission-to-Capture Ratio in *Monteburns* with Burnup to Thermal Ones Used in Ref. 7

Isotope	*Monteburns* Change in σ_f/σ_c*	Published σ_f/σ_c[7]
U-235	4.54 to 4.60	4.17
U-236	0.035 to 0.043	-
U-238	0.109 to 0.108	-
Pu-239	1.74 to 1.81	1.84
Pu-240	0.0025 to 0.006	-
Pu-241	2.68 to 2.73	2.66
Pu-242	0.012 to 0.014	-

* σ_f/σ_c is the fission-to-capture ratio of the isotope.

Another difference seen in this analysis was that the absorption cross sections used by *monteburns* were not as large as those given in the reference (see Table 4b). This was because the reference used constant, thermal cross sections for PWRs most likely at room temperature (i.e., an energy of 0.0253 eV), whereas *monteburns* calculated spectrum-dependent cross sections at actual temperatures. This higher temperature affected the cross section in two ways. First, higher temperatures cause resonances to broaden, increasing resonance absorption in the system. Second, as the temperature of the moderator increases, its density decreases, causing less of it to be present, and absorption cross sections decrease because neutrons are not slowed down as effectively. The result of these two effects is that a one-group cross section can either increase or decrease with temperature (in this case they decreased).

Table 4b. Comparison of the Change in the Absorption Cross Section in *Monteburns* with Burnup to Thermal Ones Used in Ref. 7

Isotope	*Monteburns* Change in σ_a	Published σ_a[7]
U-235	58 to 66	556
U-236	9 to 6	124
U-238	~ 1 +/- 0.05	2.23
Pu-239	~ 190 +/- 15	1620
Pu-240	363 to 92	260
Pu-241	166 to 192	1570
Pu-242	35 to 26	381

* σ_a is considered here to be the total effective microscopic absorption cross section (i.e., capture + fission) in barns (b), but in the remainder of the document, absorption solely refers to capture interactions

4.1.1.5 Fission Products

The change in relative concentrations of fission products calculated by *monteburns* matched almost identically to those produced using thermal cross sections and the equations in the reference. To model the amount of buildup of all fission products (and not just those with cross sections in MCNP), the lump sum fission

product option in MCNP was used (see Section 3.3.2 for a description). This test case showed that this option in *monteburns* did calculate fission product compositions correctly. Unfortunately, it adversely affected uranium and plutonium isotopic compositions. Either the absorption cross sections of these general fission products were too large relative to the rest of the system, or the atom densities of fission products calculated by MCNP were too large (see Section 4.2.1.3 for more information). In either case, the addition of these lump summed fission products caused the k_{eff} of the system to decrease significantly as a function of burnup. A subcritical system significantly alters the neutron energy spectrum, influencing the value of the spectrum-averaged one-group cross sections as well as the relative ratios of fission, capture, and leakage. Because this system was modeled as an infinite lattice with no leakage, simply the ratio of fission to capture was altered, causing too little U-235 to be depleted and too many plutonium isotopes to be built up. Thus, the results presented in this test case were obtained from two different runs; one to obtain actinide concentrations as a function of burnup for a near-critical system, and one to calculate the change in the total fission product concentrations with burnup.

4.1.2 Pin-Cell Burnup

The next test case compared results from *monteburns* to experimental data and results from previous calculations using other codes for a simple fuel pin within a square-pitched cell (pin-cell geometry). The Organization for Economic Cooperation and Development/Nuclear Energy Agency (OECD/NEA) Burnup Credit Calculational Criticality Benchmarks are a compendium of calculations performed by 16 different organizations (21 sets of results) and measured burnup data.[20] The purpose of these benchmarks was to determine if various computer codes/models could accurately calculate the composition of spent fuel assemblies for the Burnup Credit program. Results from Burnup Credit Benchmark Phase I-B were used in the benchmarking process for

monteburns to determine if similar nuclide concentrations were calculated as a function of burnup for a simple pin-cell geometry. The percent errors in this case were calculated relative to the measured data, and the results obtained by *monteburns* were compared to the data published by other organizations and codes.

4.1.2.1 Description

The pin-cell geometry used for this benchmark case consisted of a fuel pin (initially comprised of UO_2) with a thin layer of cladding (Zircaloy-2) surrounded by water in a square-pitched cell (see Figure 4). Reflective boundary conditions were used on all four edges to simulate that the pin was infinitely surrounded by similar pins. The parameters used for this test case are given in Table 5, and the input files used to run *monteburns* for this test case appear in Appendices C-E (which is why detailed geometry information is provided for this test case and not the others).

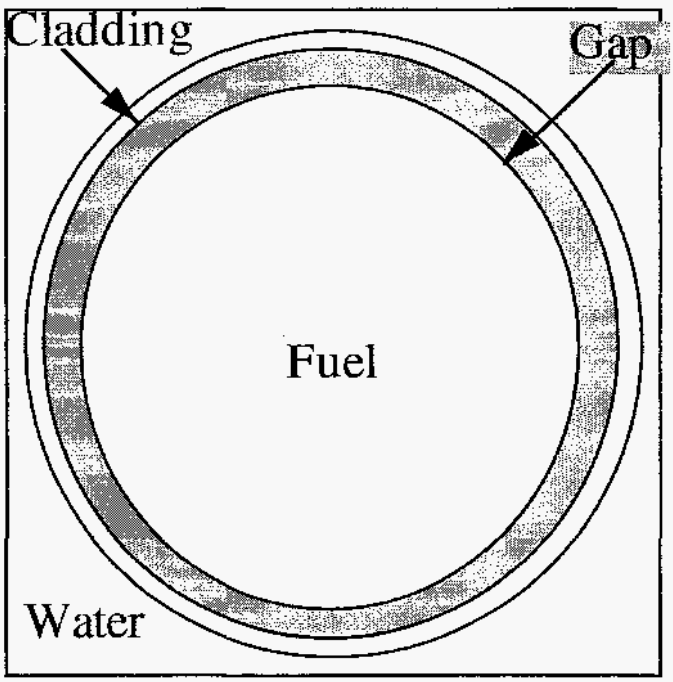


Figure 4. Pin-Cell Diagram

Table 5. Parameters for Test Case #2

Parameter[20]	Value	Parameter	Value
Fuel Density	10.045 g/cc	Length of Irradiation - Cycle 1	306 days
Water Density	0.7569 g/cc	Time Between Cycles 1 and 2	71.0 days
Rod Pitch	1.5586 cm	Length of Irradiation - Cycle 2	382 days
Rod Outside Radius	0.559 cm	Time Between Cycles 2 and 3	83.1 days
Rod Inside Radius	0.493 cm	Length of Irradiation - Cycle 3	466 days
Fuel Pellet Radius	0.4782 cm	Time Between Cycles 3 and 4	85.0 days
Active Fuel Length	347.2 cm	Length of Irradiation - Cycle 4	461 days
Effective Fuel Temp.	841 K	Length of Final Cool-Down	1870 days
Cladding Temperature	620 K	Boron Concentration - Cycle 1	331 ppm *
Water Temperature	558 K	Boron Concentration - Cycle 2	470 ppm
Ending Fuel Burnup for Scenario A (GWd/MTHM)	27.35	Boron Concentration - Cycle 3	504 ppm
Ending Fuel Burnup for Scenario B (GWd/MTHM)	37.12	Boron Concentration - Cycle 4	493 ppm

* ppm (or parts per million) means the grams of boron particles per million grams of water in the system

The soluble boron concentration in the water was fixed for each burn step and was not burned (the ability to change the composition of material in a region during burn steps without burning the material is one of the unique features of *monteburns*). If the boron were burned, the ratio of Boron-10 to Boron-11 throughout the burn step would have been affected because Boron-10 burns faster than Boron-11, and the results would not have reflected a representative neutron spectrum due to an inaccurate boron composition. In a reactor system, the coolant flows in and out of the reactor vertically and does not stay in one location for too long to be irradiated (it only takes the coolant about 0.7 seconds to flow through the reactor (see Equation 22)). Thus, it was assumed for this test case that the boron concentration going in was fixed as natural boron (about 20% B-10 and 80% B-11) and that it came out at the same concentration.

$$t = \frac{\rho * A * L}{cf} \tag{22}$$

where: t = time coolant spends in core (s) ≈ 0.7 s

ρ = density of coolant ≈ 0.7569 g/cm^3 [20]

A = cross sectional area of coolant flow ≈ 50,500 cm^2 for a PWR[4]

L = length of fuel rod = 347.2 cm [20]

cf = coolant flow ≈ 19*10^6 g/s [20]

The isotopic compositions (in mg/g initial UO_2) resulting from burnup calculations in *monteburns* for two different scenarios appear in Tables 6a and 6b. The values calculated by *monteburns*, the measured data from Ref. 20, the percent error between the two (calculated using Equation 23), and the range of values calculated by other organizations are all listed in these tables.

$$\% \text{ Error} = (\text{Calculated/Measured} - 1) * 100\% \tag{23}$$

The geometry used in this test case, an infinite lattice of fuel pins, was not completely representative of the actual system in which the measured fuel pin was burned. Thus, the main purpose of this test case was not necessarily to analyze how well it represented an actual fuel pin, but to show that *monteburns* calculated results of burnup calculations within the range of values calculated by other codes using the same geometry. The only difference between the system modeled in *monteburns* and that described in the reference is that it was difficult to obtain the exact amount of burnup in *monteburns* that was specified in the problem. This is because *monteburns* requires the user to input the total system power, irradiation time, and fraction of power produced in each step, and it then calculates how much power is generated by each region (see Equations 9-17). The resulting flux is subject to statistical errors and may not correspond

to the exact flux and power specified in the input, but this problem can be corrected by running MCNP with better statistics. The actual burnups calculated by *monteburns* for Scenarios A and B in this test case were 27.34 and 37.38 GWd/MTHM respectively, which were fairly close to the specified inputs of 27.35 and 37.12 GWd/MTHM respectively.

4.1.2.2 Results

Table 6a. Results and a Comparison of Experimental Data for Scenario A

Isotope	*Monteburns* Value (mg/g UO_2)	Published Value (mg/g UO_2) [20]	% Error	Range of Values from Other Codes [20]
U-234	0.156	0.160	-2.45	0.1330 to 0.1750
U-235	8.10	8.47	-4.32	7.445 to 8.661
U-236	3.21	3.14	2.09	3.128 to 3.540
U-238	838	843	-0.50	836.7 to 841.5
Np-237	0.286	0.268	6.65	0.2527 to 0.3396
Pu-238	0.095	0.101	-6.12	0.05721 to 0.1083
Pu-239	3.94	4.26	-7.50	3.660 to 4.690
Pu-240	1.68	1.72	-2.00	1.573 to 1.860
Pu-241	0.663	0.681	-2.72	0.5310 to 0.7335
Pu-242	0.308	0.289	6.65	0.2000 to 0.3192
Am-241	0.232	N/A	N/A	0.2269 to 0.2598
Am-243	0.0411	N/A	N/A	0.03480 to 0.04672
Mo-95	0.563	N/A	N/A	0.5590 to 0.5795
Tc-99	0.595	N/A	N/A	0.5648 to 0.6904
Cs-133	0.866	0.850	1.91	0.6820 to 0.8640
Cs-135	0.376	0.360	4.46	0.3728 to 0.3959
Nd-143	0.611	0.613	-0.36	0.6040 to 0.6792
Nd-145	0.511	0.510	0.19	0.4984 to 0.5151
Sm-147	0.160	N/A	N/A	0.1564 to 0.1932
Sm-149	0.00157	0.00290	-45.76	0.001626 to 0.002900
Sm-150	0.180	0.207	-13.22	0.1713 to 0.2146
Sm-151	0.00890	N/A	N/A	0.006376 to 0.01413
Sm-152	0.0858	0.0870	-1.35	0.07947 to 0.1073
Eu-153	0.0830	0.0790	5.11	0.06730 to 0.08921
Gd-155	0.00394	N/A	N/A	0.001507 to 0.005762

Table 6b. Results and a Comparison of Experimental Data for Scenario B

Isotope	*Monteburns* Value (mg/g UO_2)	Published Value (mg/g UO_2) [20]	% Error	Range of Values from Other Codes[20]
U-234	0.133	0.140	-5.05	0.1080 to 0.1570
U-235	4.67	5.17	-9.66	4.022 to 5.510
U-236	3.62	3.53	2.68	3.526 to 3.930
U-238	830	833	-0.28	829.2 to 836.0
Np-237	0.407	0.356	14.38	0.3560 to 0.4919
Pu-238	0.182	0.189	-3.84	0.1144 to 0.2069
Pu-239	4.03	4.36	-7.46	3.710 to 4.877
Pu-240	2.18	2.24	-2.47	1.996 to 2.347
Pu-241	0.866	0.903	-4.05	0.7510 to 0.9846
Pu-242	0.619	0.576	7.41	0.4200 to 0.6347
Am-241	0.294	N/A	N/A	0.2880 to 0.3418
Am-243	0.108	N/A	N/A	0.09637 to 0.1391
Mo-95	0.735	N/A	N/A	0.7214 to 0.7545
Tc-99	0.782	N/A	N/A	0.7327 to 0.8372
Cs-133	1.12	1.09	2.55	0.8784 to 1.117
Cs-135	0.419	0.400	4.79	0.3967 to 0.4317
Nd-143	0.711	0.716	-0.76	0.7013 to 0.8254
Nd-145	0.655	0.653	0.26	0.6326 to 0.6600
Sm-147	0.170	N/A	N/A	0.1659 to 0.2201
Sm-149	0.00164	0.00300	-45.18	0.001736 to 0.003092
Sm-150	0.247	0.271	-8.96	0.2.297 to 0.3152
Sm-151	0.00958	N/A	N/A	0.008614 to 0.01571
Sm-152	0.104	0.104	-0.20	0.09761 to 0.1416
Eu-153	0.123	0.109	13.17	0.09960 to 0.1309
Gd-155	0.00703	N/A	N/A	0.002538 to 0.01028

The results calculated by *monteburns* fell within the range of values calculated by other codes for both scenarios with the exception of the fission products Cesium (Cs)-133 and Samarium (Sm)-149. However, neither of these two isotopes' compositions were too far out of range, which means that *monteburns* represented the system just as well as or better than the other burnup codes. It is difficult to calculate fission product

concentrations accurately (as discussed in Section 4.1.2.5), so this benchmark was indeed considered to be successful.

The results of calculations performed in *monteburns* for Scenario A matched the measured results from this test case to within a 5% error for most isotopes with the exception of Neptunium (Np)-237, Pu-238, Pu-239, Pu-242, Europium (Eu)-153, Samarium (Sm)-149, and Sm-150. The errors seen in these calculations could be a result of four different effects: 1) the system, as modeled, was supercritical and produced a different spectrum than was seen experimentally, 2) the recoverable energy per fission may not have been represented correctly, 3) incorrect fission yields in ORIGEN2, and 4) statistical errors.

4.1.2.3 Differences in Energy Spectra

First, the simple system modeled in this test case was an infinite pin-cell geometry and did not represent the exact spectrum that would have been seen with a pin taken from an experimental reactor operating at steady-state. A pin in an actual reactor would be subject to the influences of other poisons (besides soluble boron) in the system, and the effects of leakage would decrease the relative reaction rates of fission and capture because it is a competing process. However, in this infinite lattice of fuel pins, no leakage occurred, so the fraction of neutrons that would have previously left the system contributed to fission and capture interactions instead. This could explain why more U-235 was depleted in *monteburns* than experimentally.

4.1.2.4 Recoverable Energy Per Fission

The second source of error could have been that the value input in *monteburns* for the recoverable energy per fission may have been too low. In *monteburns*, the user has the option to input the recoverable energy per fission for U-235, and the actual recoverable energy per fission (Q_{fis}) in the system is scaled relative to the presence of

other actinides in the system and the ratio of their recoverable energies to that of U-235 (see Equations 5-7). For this case, an estimated value of 200 MeV was used.[16] Thus, because only 200 MeV was generated per fission, a larger number of fissions were required for a given power level than if a larger value, such as 202 MeV, were entered. For example, results from using 202 MeV showed that U-235 was not burned as quickly because fewer total fissions occurred. However, the fission-to-capture ratio of each actinide remained the same even though the recoverable energy per fission changed, so the plutonium isotopes still did not build up as much as in the measured data. In either case, it was difficult to justify using a higher recoverable energy per fission in this test case without experimentally showing that a PWR system provides that much more energy per fission.

4.1.2.5 Fission Yields

Third, for both scenarios the compositions of Sm-149, Sm-150, and Eu-153 calculated by *monteburns* at the end of the irradiation were smaller than measured results (by almost a factor of 2 for Sm-149 although much better for the other two). This is probably a result of estimations of the fission yields made by ORIGEN2 for these isotopes. For example, the total fission yield from Pu-239 for Sm is around 0.2% in the ORIGEN2 libraries while it is 0.7% experimentally,[21] causing fewer Sm atoms to be produced in *monteburns* than experimentally. However, this is a facet of the code ORIGEN2 and cannot currently be modified by *monteburns*, so these errors must be accepted. Additionally, the ratio of fissions due to Pu-239 versus Pu-241 may also have affected the results. The fission yields of both Sm-149 and Sm-150 are slightly greater from Pu-239 than Pu-241 according to the relevant ORIGEN2 library. Because excess Pu-242 was produced, it was assumed that a great deal of Pu-241 was also produced (although it was depleted rather quickly). Thus, more fissions probably occurred from Pu-241 than Pu-239 in the modeled system than the measured one, and fewer Sm-149 and

Sm-150 atoms were produced. The errors associated with Eu-153 and other fission products were probably a result of similar reasons. Fortunately, ORIGEN2 contains more representative fission yields for a majority of the fission products, overall producing acceptable results (i.e. < 5% error).

4.1.2.6 Statistical Variances

The last possible source of error could have been a result of the statistical variances involved with obtaining spectrum-averaged, one-group cross sections in MCNP, which were produced using tally cards. The accuracy of these depend on the statistics with which MCNP was run and the accuracy with which it calculates fluxes in each region. For example, Pu-238 displayed a 6.12% error in Scenario A but only a 3.84% error in Scenario B. The accuracy to which the ENDF/B-V cross section set(s) represented resonances may also have affected the outcome. Either way, variances in cross sections may have altered the amount of resonance absorption versus self-shielding and influenced results.

4.1.2.7 Additional Burnup

For Scenario B, a number of additional actinides had errors greater than 5%. This case involved higher burnups than Scenario A as well as a larger variance between the final burnup in *monteburns* and measured data, so greater percent errors were expected. This was because variances in cross sections and fission yields became more prominent as power times time increased because each burn step became relatively longer (in terms of GWd) to make differences more prominent. In this scenario of the test case, it was particularly obvious that U-235 was burned faster using *monteburns* than experimentally, creating almost a 10% error. This was again due to the reasons discussed previously. However, all actinides still fell within the range of computational values produced by

other codes for both scenarios, showing the validity of *monteburns* in modeling the system described by the reference.

Overall, the ability of *monteburns* to calculate the change in composition of a system with burnup has been shown to be fairly good and within the range of values calculated by other codes. More accurate answers may be obtained using better statistics (as further discussed in Section 4.2) or by modeling the entire system rather than just one fuel pin to represent a more accurate spectrum and to include the effects of leakage.

4.1.3 Assembly Burnup

The purpose of the third test case was to compare the burnup results calculated by *monteburns* to experimental values for a full PWR assembly.

4.1.3.1 Description

The assembly modeled in this example was H.B. Robinson's Unit 2, which uses a Westinghouse 15x15 fuel lattice, and the assembly layout is shown in Figure 5 (for detailed information, see Ref. 19). This test case studied four different scenarios, each with a different final burnup. To simulate an assembly located in the middle of a reactor with identical assemblies surrounding it, reflective boundary conditions were placed on all four sides of the assembly.

This model was considered to be more accurate than the simple pin-cell one in Test Case #2 because burnable poisons as well as guide and instrumentation tubes were represented, thus, the spectrum of the system should have been more accurate. However, the same number of outer burn steps were used for each scenario with increasing amounts of power times time, so representative cross sections were calculated over a shorter time frame in the first scenario and over a longer one in the last one. The same average boron concentration was also used for each but probably represented the middle two cases best.

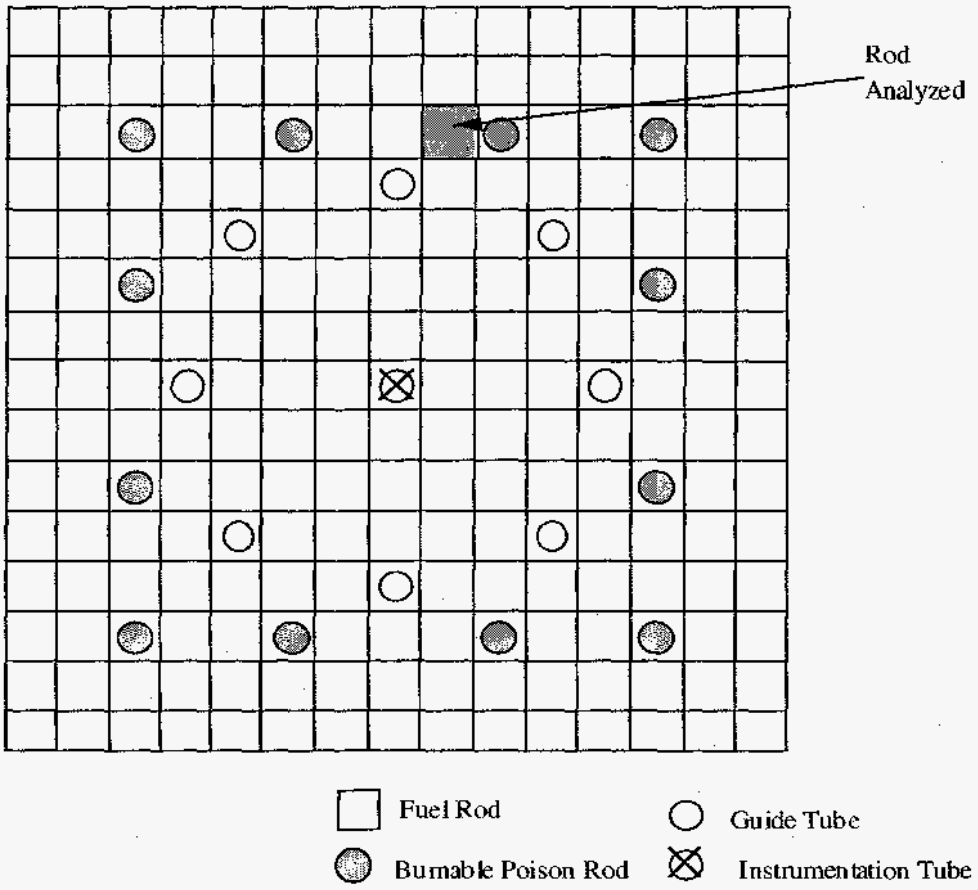


Figure 5. Layout of Assembly for Test Case #3

The system turned out to be slightly supercritical for the first scenario and slightly subcritical for the last one, so the results for the middle two cases were expected to be better than for the first and last. Again, there were difficulties achieving the exact amount of burnup specified in the input, but the values were fairly close nonetheless (16.00, 23.84, 28.64, and 31.86 GWd/MTHM compared to 16.02, 23.81, 28.47, and 31.66 GWd/MTHM for Scenarios 1-4 respectively).

4.1.3.2 Results

One rod within this assembly was measured for isotopic content, and the measured results for this rod were compared to those calculated by *monteburns* in Tables

7a and 7b (in g/g UO_2) for the four burnup scenarios. The percent errors displayed in these tables were calculated using Equation 23.

Table 7a. Results for Burnups of 16.00 and 23.84 GWd/MTHM (g/g UO_2)

Burnup	16.00 GWd/MTHM			23.84 GWd/MTHM		
Isotope	*monteburns*	published[19]	% error	*monteburns*	published[19]	% error
U-235	0.0110	0.0107	2.62	0.00751	0.00721	4.11
U-236	0.00212	0.00219	-3.37	0.00266	0.00274	-3.09
U-238	0.848	0.847	0.17	0.842	0.847	-0.53
Pu-238	2.89E-05	2.83E-05	2.29	7.01E-05	6.95E-05	0.83
Pu-239	0.00371	0.00364	2.01	0.00407	0.00402	1.31
Pu-240	0.00114	0.00109	4.22	0.00170	0.00167	1.61
Pu-241	3.25E-04	3.04E-04	7.04	5.29E-04	5.04E-04	4.97
Np-237	1.51E-04	1.55E-04	-2.76	2.46E-04	2.60E-04	-5.55
Tc-99*	6.06E-06	5.44E-06	11.35	8.76E-06	8.09E-06	8.34
Cs-137*	0.0353	0.0359	-1.64	0.0527	0.0539	-2.22

Table 7b. Results for Burnups of 28.64 and 31.86 GWd/MTHM (g/g UO_2)

Burnup	28.64 GWd/MTHM			31.86 GWd/MTHM		
Isotope	*monteburns*	published[19]	% error	*monteburns*	published[19]	% error
U-235	0.00603	0.00618	-2.44	0.00515	0.00486	5.98
U-236	0.00285	0.00282	1.24	0.00295	0.00300	-1.51
U-238	0.838	0.834	0.54	0.835	0.842	-0.89
Pu-238	1.06E-04	1.14E-04	-7.01	1.33E-04	1.30E-04	1.97
Pu-239	0.00431	0.00439	-1.77	0.00445	0.00420	6.00
Pu-240	0.00199	0.00197	1.14	0.00218	0.00212	2.65
Pu-241	6.49E-04	6.81E-04	-4.72	7.11E-04	6.92E-04	2.71
Np-237	3.11E-04	3.04E-04	2.45	3.59E-04	3.33E-04	7.91
Tc-99*	1.03E-05	8.95E-06	14.94	1.13E-05	1.01E-05	11.90
Cs-137*	0.0631	0.0627	0.70	0.0703	0.0713	-1.44

*The units for these are given in Curies/gram UO_2 (Ci/g) instead of g/g UO_2 like the other isotopes.

As can be seen from these tables, the percent error associated with a majority of the isotopes in these cases was below 5% with the exception of several actinides and the fission product Technetium (Tc)-99.

4.1.3.3 Actinides

In each burnup case, at least one actinide concentration resulted in a percent error greater than 5%, but none consistently produced poor results. These errors were probably a result of any or all of the reasons presented in Test Cases 1 and 2 (i.e., resonance self-shielding, cross sections, inaccurate system modeling, variances in recoverable energy per fission, statistics, etc.). Because the first scenario was slightly supercritical and the last subcritical, the spectrums were probably not representative of a steady-state system, and cross sections may have suffered accuracy as a result. This is probably a result of differences in the locations of resonances and the amount of resonance absorption versus self-shielding that occurred. For example, in Scenarios 1, 2, and 4, too much U-238 was depleted, producing excess Pu-239, and in Scenario 3, too little U-238 was depleted, not producing enough Pu-239 or higher plutonium isotopes. In contrast, too much U-235 was depleted in Scenario 3 because Pu-239 did not contribute to as many fissions as it should have, and excess U-236 was produced. In turn, not enough U-235 was depleted in Scenarios 1, 2, and 4 because too much Pu-239 and Pu-241 fissioned, resulting in too little production of U-236. This probably means that in Scenario 3, the absorption cross section of U-235 was too large compared to that of U-238, whereas in the other test cases, it was too small. Thus, the number of U-235 captures appeared to be indirectly proportional to the number of U-238 captures in this test case, and in all scenarios were slightly different than the actual system.

4.1.3.4 Fission Products

The percent errors associated with the concentration of Tc-99 were around 10-15 percent for each burnup case. There are three potential sources of error for this calculation. First, the fission yields for Tc-99 used by ORIGEN2 may not have been truly representative of the probability that it was produced by fission (as discussed in Section 4.1.2.5). Second, the absorption cross section calculated by *monteburns* for Tc-99 may have been too small because not enough of it was transmuted to Tc-100. Finally (but least likely), the concentration of Tc-99 was given in Ci/g UO_2 instead of g/g UO_2 as the actinides were, and the conversion may have been performed incorrectly. *Monteburns* outputs the concentrations of isotopes in grams, so it was converted from grams to Curies by multiplying by the specific activity of Tc-99 (see Equation 24 [22]).

$$SA = \frac{4.17 * 10^{23}}{MT} \sim 1.7\text{e-}2 \text{ Ci/g for Tc-99} \qquad (24)$$

where: SA = specific activity (Bq/g) (where 1 Ci = $3.7*10^{10}$ Bq [22])

M = atomic weight of isotope ≈ atomic mass number ≈ 99 for Tc-99

T = half-life of isotope in seconds = $2.13*10^{5}$ years [21]

However, the errors associated with the fission product Cs-137 were less than 2.5% using the same ORIGEN2 library and specific activity equation. Therefore, the errors associated with Tc-99 were more likely a result of the differences in the fission yields or cross sections. Even a 10-15% error for a fission product was not considered to be too unreasonable in this analysis considering all the uncertainties and potential statistical errors involved.

4.1.3.5 Comparison to SCALE

The percent errors seen using the code SCALE were similar to those obtained from *monteburns*. For Tc-99 the average percent error was 11.7% in SCALE and between 8-15% in *monteburns*. Similarly, the errors associated with the other fission product, Cs-137, were only on average, 1.2% in SCALE and between 0.7 and 2.5% in *monteburns*. This means that the two codes produced similar results, which is probably because the same (or similar) fission yields and/or cross sections were used in each (this is because ORIGEN-S, the code used by SCALE containing fission yields, is simply a newer version of ORIGEN2, which is used in *monteburns*) as well as the same model. The largest percent errors seen in SCALE for actinides were associated with Pu-239, Pu-241, and Np-237 (8.2%, 5.4%, and 11.1% respectively) for this test case, and comparing these to Tables 7a and 7b, *monteburns* performed as well as SCALE for burnup calculations. A more accurate system model would be needed to match measured results more closely.

Overall, modeling a full reactor assembly proved to be more accurate than just modeling an infinite lattice of identical fuel pins, and it was shown that *monteburns* performs calculations for a given system model just as well as a code such as SCALE.

4.1.4 Power Distribution

One of the many capabilities of *monteburns* is that it can calculate the amount of power produced in each region/material of a system given the total system power. Power distribution is important because it determines how much energy is released from each region, thereby indicating which one(s) is depleted the fastest. It does this by obtaining the flux and macroscopic fission cross section tallies for the region(s) of interest from MCNP, "normalizing" these values, and then calculating the power in each region from these results (see Equations 9-17 for more information).

4.1.4.1 Description

The test case used to validate this calculation modeled a sample 3x3 BWR assembly with eight fuel pins on the outside and a rod capable of containing burnable poison in the middle.[23] The layout of the 3x3 assembly is shown in Figure 6, and the pins are numbered according to three different regions. The average power produced per pin in the assembly was calculated, and then the power produced by a pin in each region was divided by this average.

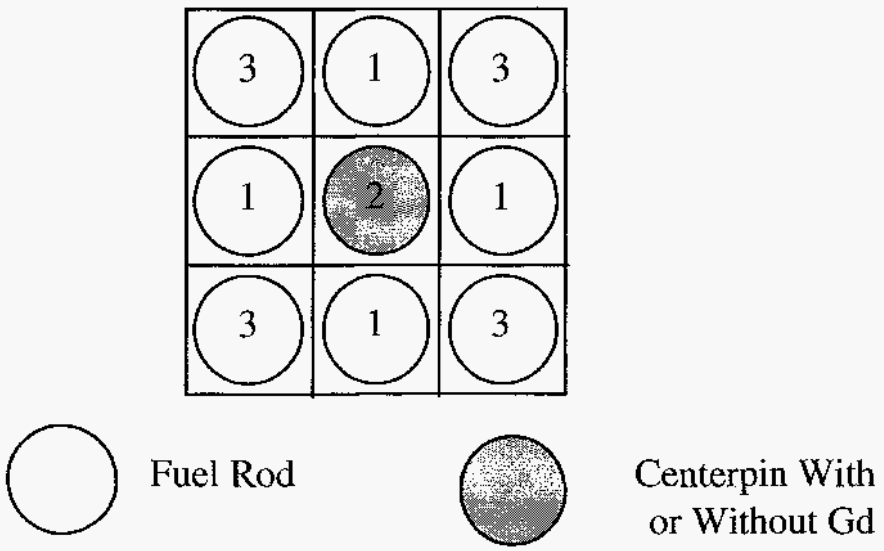


Figure 6. 3x3 Assembly

4.1.4.2 Results

Table 8 displays the differences between the results calculated by *monteburns* and the range of values obtained using other codes given in Ref. 23 for both a scenario with gadolinium (Gd) in the center pin and one without. This table shows that the power distributions for both cases fell within the range of published values, indicating that not only does *monteburns* perform power distribution calculations correctly, but it also analyzes a BWR fuel assembly well.

Table 8. Pin Power Distribution

Fuel Pin Analyzed	*Monteburns* Value*	Published Range of Values [23]
Pin 1 with 3% Gd	1.055	1.053 to 1.062
Pin 2 with 3% Gd	0.437	0.413 to 0.460
Pin 3 with 3% Gd	1.086	1.082 to 1.087
Pin 1 with 0% Gd	1.031	1.029 to 1.032
Pin 2 with 0% Gd	0.766	0.766 to 0.779
Pin 3 with 0% Gd	1.028	1.026 to 1.027

*This is the average power produced per pin in each region divided by the average power produced per pin in the 3x3 assembly.

Additionally, it shows that the continuous pointwise cross sections output as one-group in MCNP produce compatible results to the group-wise ones used by the other codes in this reference.[23]

4.1.5 Activity Calculation

One of the proposed future uses of *monteburns* is to provide activation and/or decay powers of materials (see Section 6.0). To do this, the activities of various isotopes in a material must be calculated. This test case compares the activity of a spent fuel assembly containing MOX fuel after irradiation in *monteburns* to published results from SCALE. The purpose of using MOX fuel in this test case was to show the versatility of *monteburns* in calculating the burnup of plutonium- as well as uranium-based fuels.

4.1.5.1 Description

First, the composition of the material after irradiation was calculated using *monteburns*, and then it was converted and output as activity as a function of decay time using ORIGEN2 (although only the activity immediately after removal is compared here). This information can be used to generate dose rates as a function of cooling time for a spent fuel assembly, which could be useful in both repository analyses and proliferation issues.

4.1.5.2 Results

The differences between *monteburns* and published values using SCALE are shown in Table 9 for a Combustion Engineering System 80+ PWR System containing mixed-oxide (MOX) fuel.[24] The percent difference between the values calculated by *monteburns* and those given in Ref. 24 for SCALE were calculated by Equation 23, where the measured value was replaced by the SCALE value.

Table 9. Results from Activity Calculation

Activity (Ci)	*Monteburns*	SCALE[24]	% difference
H-3	3.48E+02	2.76E+02	26.09
Kr-85	2.85E+03	2.69E+03	5.99
Kr-85m	5.25E+04	5.07E+04	3.47
Rb-86	4.91E+02	3.72E+02	31.88
Kr-88	1.27E+05	1.30E+05	-2.08
Sr-89	1.66E+05	1.69E+05	-1.66
Sr-90	1.94E+04	2.00E+04	-2.95
Y-90	2.00E+04	2.03E+04	-1.38
Sr-91	2.35E+05	2.45E+05	-4.24
Y-91	2.39E+05	2.46E+05	-3.05
Y-91m	1.36E+05	1.42E+05	-4.01
Sr-92	2.80E+05	2.90E+05	-3.59
Y-92	2.82E+05	2.91E+05	-3.13
Y-93	3.60E+05	2.45E+05	46.86
Nb-95	4.54E+05	4.56E+05	-0.44
Nb-95m	3.21E+03	5.21E+03	-38.35
Zr-95	4.52E+05	4.58E+05	-1.35
Zr-97	4.93E+05	4.86E+05	1.42
Mo-99	5.65E+05	5.89E+05	-4.13
Tc-99m	4.99E+05	5.22E+05	-4.35
Rh-105	4.41E+05	4.99E+05	-11.64
Rh-105m	1.39E+05	1.45E+05	-4.21
Ru-105	4.96E+05	5.10E+05	-2.71
Ru-106	3.76E+05	3.74E+05	0.61
Sb-127	4.52E+04	3.86E+04	17.10

Table 9 (cont.)

Activity (Ci)	*Monteburns*	SCALE[24]	% difference
Te-127	4.61E+04	3.84E+04	20.03
Te-127m	6.40E+03	6.69E+03	-4.33
Sb-129	1.24E+05	1.22E+05	1.97
Te-129	1.23E+05	1.17E+05	4.87
Te-129m	1.86E+04	2.41E+04	-22.86
I-131	3.39E+05	3.43E+05	-1.28
Xe-131m	3.80E+03	4.47E+03	-14.92
I-132	4.78E+05	4.90E+05	-2.45
Te-132	4.65E+05	4.76E+05	-2.29
I-133	6.31E+05	6.52E+05	-3.22
Xe-133	6.02E+05	6.55E+05	-8.05
Xe-133m	1.89E+04	2.16E+04	-12.50
Cs-134	8.41E+04	7.03E+04	19.59
Cs-134m	2.13E+04	1.31E+04	62.75
I-135	5.90E+05	6.24E+05	-5.40
Xe-135	5.12E+05	4.80E+05	6.56
Xe-135m	1.38E+05	1.52E+05	-9.01
Cs-136	4.70E+04	4.17E+04	12.71
Cs-137	5.53E+04	5.70E+04	-2.98
Ba-140	5.13E+05	5.44E+05	-5.64
La-140	5.32E+05	5.55E+05	-4.14
Ce-141	4.90E+05	5.00E+05	-2.08
La-141	4.86E+05	4.96E+05	-2.08
Ce-143	4.19E+05	4.29E+05	-2.28
Pr-143	4.19E+05	4.21E+05	-0.40
Ce-144	3.51E+05	3.51E+05	-0.11
Nd-147	1.99E+05	2.05E+05	-3.12
Np-238	2.23E+04	3.48E+04	-35.98
Pu-238	1.29E+03	1.24E+03	4.27
Pu-239	9.15E+02	8.93E+02	2.41
Pu-240	1.11E+03	1.10E+03	0.55
Am-241	6.07E+02	6.96E+02	-12.83
Pu-241	2.74E+05	2.76E+05	-0.91
Cm-242	8.37E+04	8.05E+04	3.93
Cm-244	4.68E+03	2.87E+03	62.96

Table 9 shows that the percent differences associated with most of the actinides (with the exception of Np-238, Am-241 and Cm-244) were less than 5%, but they were larger for some of the fission products.

4.1.5.3 Actinides

The percent differences seen for all plutonium isotopes and most other actinides were less than 5% (excluding Np-238, Am-241, and Cm-244), showing the validity of both codes in performing burnup calculations involving major system isotopes in the given geometry. Because this test case was not compared to experimental data, the causes of error discussed in Test Cases 1-3 were minimal here. Instead, errors associated with Np-238, Am-241, and Cm-244 were most likely due to variances in cross sections and the ways the codes model an assembly with reflective boundary conditions. SCALE uses multi-group cross section sets, whereas *monteburns* uses one-group spectrum-averaged ones obtained from continuous-energy data in MCNP. SCALE also typically uses the Monte Carlo code KENO, whereas *monteburns* uses MCNP. Additionally, even though results from the two codes were comparable, they may not complement measured data as well without a better system model.

The Am-241 concentration in *monteburns* was probably smaller than that in SCALE because not enough Pu-241 was present to decay by beta emission to Am-241, which was probably a result of fewer neutron absorptions in Pu-240. Another explanation could be that the Am-241 absorption cross section was larger in *monteburns* than in SCALE, producing higher actinide concentrations while depleting Am-241. This explanation is probably more likely because the *monteburns* concentrations for Cm-242 and Cm-244 were larger than those in SCALE. By the absorption of a neutron, Am-241 is transmuted to Am-242, which beta decays to Cm-242; Cm-242 then absorbs neutrons to create Cm-244. The small concentration of Am-241 in *monteburns* relative to SCALE also contributed to the relatively small concentration of Np-238 (Am-241 decays by

alpha emission into Np-237, which absorbs a neutron to become Np-238). As the Am-241 concentration was relatively low in *monteburns*, the resulting decay process produced less Np-237, and in turn, fewer Np-238 atoms.

4.1.5.4 Fission Products

Fission products with a deviation greater than 5% between SCALE and *monteburns* include: Ba-140, Cs-134, Cs-134m, Cs-136, H-3, I-135, Kr-85, Nb-95m, Rb-86, Rh-105, Sb-127, Te-127, Te-129m, Xe-131m, Xe-133, Xe-133m, Xe-135, Xe-135m, and Y-93. From a list of 53 different fission products, having only 19 with a percent difference over 5% and only 13 greater than 10% is pretty good. This means that *monteburns* calculated almost 75% of all fission product concentrations fairly well (less than 10% difference) in comparison to SCALE and about two-thirds of them to a less than 5% difference. The deviations seen with these fission products were probably due to fission yield and/or cross section variances between the two codes. Thus, having relatively good results for 75% of the fission products was considered to be acceptable.

Overall, the results obtained using *monteburns* were fairly close to those expected for each test case, and a majority of them were within a relative error/difference of 5% of measured results. Almost all were within the range(s) of published calculations from other codes. First, the change in relative concentrations of uranium and plutonium isotopes were comparable to those referenced.[7] Next, a full assembly model was shown to produce better results than a pin-cell geometry due to a more accurate spectrum representation. Finally, more similarities were found when comparing results from *monteburns* to calculations performed with another code (such as SCALE) using the same geometry/model than comparing to measured results from a rod irradiated in a full reactor system influenced by leakage, interfacing between assemblies, and other features.

Both PWR and BWR cases were tested in *monteburns*, along with both uranium- and plutonium-based fuels. The technique used in *monteburns* for generating cross sections differed from what other codes such as SCALE use (i.e., one-group spectrum-averaged ones obtained from continuous energy data versus multi-group ones), but the differences between the two did not appear to be significant. Thus, *monteburns* was considered adequate for the problems presented here. Unfortunately, there is not currently any readily available experimental data for a fast system, such as that used in ATW, so no benchmarks were performed for one. However, it is assumed that since the code has been shown to work well for a thermal system, it can calculate decent results for a fast system as well.

4.2 Statistical Analyses

Another important aspect of developing and/or running a computer code is to determine how statistics affect the results. The term statistics, when used in reference to *monteburns*, refers to how results vary using different input parameters or modeling a system in different ways. To test this variance, several of the test cases discussed in the previous section were further examined. No MCNP statistical runs are presented here; many of these have already been performed by others in the industry (for example, Ref. 11).

4.2.1 Input Parameters

The input parameters analyzed for their effect(s) on statistics were: the number of outer burn steps, the number of internal burn steps, the number of predictor steps, the importance fraction, and the recoverable energy per fission. The majority of tables in this section show both the measured and calculated values for Scenario 1 of Test Case #3 at a burnup of ~ 16 GWd/MTHM for four different isotopes: U-235, U-236, Pu-239, and Pu-240. Unless otherwise stated, the number of internal burn steps was 80, the number

of outer burn steps was 8 (4 irradiation, 4 decay), the number of predictor steps was one, the importance fraction was 0.01, the U-235 recoverable energy per fission was 200 MeV, the number of neutrons per cycle was 1000, the number of active cycles was 100, and the number of skipped cycles in MCNP was 15.

4.2.1.1 Number of Outer and Internal Burn Steps

The first parameter a user typically wants to determine in *monteburns* is the length of the time intervals over which irradiation occurs. There are two input parameters that can affect this length of time: the number of outer burn steps, and the number of internal burn steps. First, using more outer burn steps not only decreases the length of each time interval but also increases the accuracy of the system because the spectrum-averaged one-group cross sections for the system are updated more frequently (consequently increasing the run time). Second, the way to use shorter time steps in ORIGEN2 without having to perform additional MCNP runs is through the use of internal burn steps. The more internal burn steps used, the shorter the time intervals for each ORIGEN2 irradiation. As discussed in Section 2.2, this is important because ORIGEN2 performs different calculations (i.e., the Bateman equations versus the matrix exponential method) for isotopes with half-lives less than 10% of the time interval.[6] Thus, using shorter time intervals may provide more accurate results for the problem. The optimum number of internal burn steps should also depend upon whether continuous or discrete (all at one time) feed is used. By using continuous feed with different beginning and ending feed rates, it was assumed when designing *monteburns* that it would be necessary to break the time steps in ORIGEN2 into even shorter periods. This is because the amount of feed added during each internal burn step is interpolated from the beginning and ending feed rates for that outer burn step and averaged over each internal burn step.

For the first scenario of test case #3 (a discrete feed case), the effects of the number of outer and internal burn steps on the results are shown in Tables 10a and 10b (these were performed with forty internal burn steps and eight outer burn steps respectively). Results for a continuous feed case (representing ATW, which will be discussed in Section 5.1) are then displayed in Table 10c.

Table 10a. Comparison of Results as a Function of Number of Outer Burn Steps

Experimental Results	(grams/ assembly)	5360	1097	1823	546
# of Outer Burn Steps	Length of ORIGEN2 steps (days)	U-235	U-236	Pu-239	Pu-240
8	6.09	5500	1060	1860	569
16	3.04	5530	1060	1880	571
24	2.03	5500	1060	1870	574

Table 10b. Comparison of Results as a Function of Number of Internal Burn Steps

Experimental Results	(grams/ assembly)	5360	1097	1823	546
# of Internal Burn Steps	Length of ORIGEN2 steps (days)	U-235	U-236	Pu-239	Pu-240
2	121.75	5510	1060	1860	559
4	60.88	5510	1060	1860	561
6	40.58	5510	1070	1860	569
8	30.44	5500	1070	1860	562
10	24.35	5540	1060	1850	565
20	12.18	5530	1060	1860	568
30	8.12	5520	1070	1880	575
40	6.09	5560	1060	1890	581
50	4.87	5500	1070	1860	574

Table 10c. Results as a Function of Internal Burn Step for Continuous Feed
(grams)

# of Internal Burn Steps	Length of ORIGEN2 steps (days)	U-238	Pu-239	Pu-240	Am-241
10	12.2	7.97E+2	1.70E+5	2.08E+5	1.79E+4
20	6.09	7.93E+2	1.68E+5	2.07E+5	1.76E+4
30	4.06	7.97E+2	1.71E+5	2.07E+5	1.79E+4
40	3.04	7.95E+2	1.70E+5	2.08E+5	1.79E+4

Surprisingly, all three of these tables show little increase in accuracy with more than the minimum required number of outer or internal burn steps (i.e. two for discrete feed and ten for continuous feed[18]) for these sample test cases. The number of outer burn steps is thus recommended to be the lowest needed to represent all system changes. For example, in this case, eight were required because there were four irradiation cycles with different amounts of power and soluble boron as well as a cooling period following each. It also appeared that using only two internal burn steps for the discrete feed case with a thermal spectrum (with an irradiation period of about 120 days) and using ten for the continuous case with a fast spectrum (corresponding to a length of approximately 12 days each) produced as good of results as using more. Thus, for similar cases to those presented here, it is recommended to use the minimum number of internal burn steps even though using additional internal burn steps does not significantly affect the run time.

Additionally, this test case at least showed that the results obtained from *monteburns* for both a fast and thermal spectrum were consistent if not influenced by changes in the number of burn steps. Nonetheless, the user should verify that the number of burn steps used provides enough accuracy for his/her specific system and associated irradiation periods. This is because ORIGEN2 may still produce poor results for irradiation periods greater than 125 days (the maximum studied here was 121.75 days) or for other types of systems or problems (such as decay-only over thousands of years).

4.2.1.2 Number of Predictor Steps

The next parameter analyzed was the number of predictor steps. For each predictor step during each outer burn step (with the exception of the first step, in which case an extra predictor step is run - see Section 3.3.2), MCNP is run to obtain one-group spectrum-averaged cross sections. Thus, increasing the number of predictor steps increases the degree to which the cross sections calculated by MCNP represent the average system spectrum for the step, but it also increases the run time of the problem. The results from this analysis appear in Table 10d.

Table 10d. Comparison of Results as a Function of Number of Predictor Steps
(grams/assembly)

Experimental Results	5360	1097	1823	546
Predictor Steps	U-235	U-236	Pu-239	Pu-240
0	5540	1050	1870	450
1	5500	1060	1860	569
2	5500	1060	1870	580

With eight outer burn steps and eighty internal burn steps, a large difference was seen between using zero and one predictor step because cross sections were calculated only once in the former case (i.e., only for the first step) and nine times in the latter. This indicates that it is indeed important to calculate cross sections several times throughout an irradiation. However, the difference between using one and two predictor steps was minimal, meaning that the one-group spectrum-averaged cross sections calculated with one predictor step were fairly good representations of the system at each step. Because the run time significantly increases with each predictor step, it was found that for this system and others studied thus far, there is no advantage in using more than one predictor step per outer burn step. Again, the differences may have become more definitive if a

case with a longer time interval and/or fewer required outer burn steps had been studied (however, one was not used because experimental data for such a system was not readily available). Either way, the user is advised to make sure that one predictor step is adequate enough for his/her system by comparing the flux spectrum and isotopic compositions (in mass) halfway through each predictor and actual step to obtain the best results.

4.2.1.3 Importance Fraction

Another input parameter varied in this statistical analysis was the importance fraction. This effectively selects which fission products are passed back to MCNP from ORIGEN2. If this value is positive, then individual fission products are passed back to MCNP (assuming their cross sections exist), allowing temperature- and system-dependent parameters to influence these individual fission product cross sections. If this value is negative, fission products produced in ORIGEN2 are added together as a total mass and sent back to MCNP as one of two general fission product representations (those from U-235 and those from Pu-239) at room temperature (see Section 3.3.2). In this case individual fission product cross sections in ORIGEN2 are not updated because only general lumped sum ones are used in MCNP and cannot effectively replace individual ones in ORIGEN2. Results from this statistical analysis appear in Table 10e.

The lower the value of the importance fraction, if positive, the smaller a contribution an isotope has to make to the system in either absorption or fission interactions, mass, or atom density (see Equations 18-21) to be included in MCNP. Surprisingly, the most accurate results for this analysis occurred when the importance fraction was relatively large (0.1 or 1.0). This is because a steady-state spectrum was best represented in these cases. The system[19] was initially modeled near critical, and as the number of fission products added to the system increased (i.e., a lower importance fraction), keff decreased because the fission products absorbed many neutrons that would

Table 10e. Comparison of Results as a Function of Importance Fraction
(grams/assembly)

Experimental Results	5360	1097	1823	546
Importance fraction	U-235	U-236	Pu-239	Pu-240
1	5480	1060	1820	560
0.1	5470	1060	1810	565
0.001	5550	1060	1900	565
0.00001	5520	1070	1910	573
-0.1	6690	1050	4340	539
-0.01	6690	1050	4340	554

have otherwise contributed to fission. Thus, the spectrum and/or cross sections were no longer representative of the system at steady-state. If this case could have been modeled more accurately (i.e., include leakage and interaction with the sides of the reactor core), then as more fission products were added to the system, then the spectrum would have been more accurate and better results would have been obtained (to represent what actually occurs in a reactor).

In this analysis the lump sum option for fission products (i.e. a negative fractional importance) produced poor results. This lump sum option in *monteburns* means that all fission products are combined into two general representations, homogenizing an otherwise heterogeneous combination of fission products. It produced poor results because the general fission product cross sections in MCNP appear to have either relatively large absorption cross sections or large atom densities compared to the case(s) where fission products are assessed individually in MCNP. As the mass of summed fission products increased with burnup, the absorption and fission interactions that occurred in U-235 and Pu-239 in MCNP decreased because too many neutrons were absorbed by the lump fission products instead. Additionally, more U-238 was transmuted to Pu-239 than should have been. This may have been because absorption

resonances exist at slightly larger energies for U-238 than U-235 and Pu-239 (above 10^{-4} MeV[25]) and many neutrons were absorbed there instead of in resonances at lower energies (this could be due to resonance self-shielding, less available moderation to slow neutrons down, and/or a shift in the energy spectrum of the system). Fewer neutrons existed in the resonance regions of U-235 and Pu-239 compared to U-238, so their one-group absorption cross sections decreased and less U-236 and Pu-240 was formed. In contrast, Pu-240, which also has absorption resonances in this higher energy range, was transmuted more quickly than in the case of individual fission products (i.e., a positive importance fraction).

The addition of fission products in the actual steady-state system also induces the effects discussed above, but the general fission product representations in MCNP seemed to exaggerate it. There are potentially two main explanations for this poor representation: the effective absorption cross sections of these two general fission products were too large relative to others in the system being studied, or the atomic weights used by MCNP to convert the weight percents obtained by *monteburns* into atom densities for Monte Carlo calculations were too small. The latter would occur if the average weights of fission products produced by ORIGEN2 were larger than the representative ones in MCNP, causing the atom density of fission products to be too large and too much absorption to occur (atom density is inversely proportional to atomic weight). Upon examination, the total weight of fission products with an atomic mass above 117 (the weight of Pu-239 general fission products) was about 1.5 times that of fission products with atomic masses below 115 (the weight of U-235 general fission products), whereas more than half of the fissions occurred from U-235. This probably resulted from the fact that many higher actinides (such as Pu-241, americium, etc.) fissioned along with U-235 and Pu-239, producing fission products with larger atomic weights than those representative of Pu-239 (which is what they lumped together as). The ending result was that the atomic weight of the representative fission product for Pu-239 was too small and the atom

density of this fission product was too large, adversely affecting the spectrum of the system. In addition, using the lump sum option does not allow individual fission product cross sections to be modified in ORIGEN2, also decreasing the accuracy of the calculations. Overall, the user is not recommended to use the lump sum option for a reactor system unless he/she completely understands the implications.

Additionally, the only effect of a negative importance fraction is in determining the contribution that actinides must make to the system to be passed back to MCNP (i.e., individual fission products are no longer included in the MCNP input file because a lump sum is used instead). The results for U-235, U-236, and Pu-239 were not affected when the importance fraction went from negative 0.1 to negative 0.01, but those for Pu-240 were affected, most likely because additional actinides were included in MCNP. Such an increase was also seen for Pu-240 as more actinides were added to the system with a positive fraction importance (at least from 1 to 0.1 and 0.001 to 0.0001). This increase was not seen between 0.1 and 0.001, probably due to statistics.

4.2.1.4 Recoverable Energy Per Fission

The last input parameter varied in this statistical analysis was the value of the recoverable energy per fission (Q_{fis}) input by the user for the actinide U-235. The input value of Q_{fis} was varied between 190 and 210 MeV, and the value of Q_{fis} calculated by *monteburns* at the end of the irradiation period was about 4 MeV greater than the input value (see Table 10f for results) due to the contribution of other actinides in the system. The number of fissions that occur in a system are determined by the required power level of the system and the value of Q_{fis}. The more energy released by each fission (i.e., the larger Q_{fis} is), the fewer fissions that must occur to meet the overall power requirement. This means that the amount of material burned is lower, causing the final concentration of fissile material initially in the system (i.e., U-235) to increase proportionally with the value of Q_{fis}.

Table 10f. Results as a Function of Recoverable Energy Per Fission (g/g UO_2)
(grams/assembly)

Experimental Results	5360	1097	1823	546	-
Input Q_{fis} (MeV)	U-235	U-236	Pu-239	Pu-240	Ending Q_{fis} (MeV)
190	5310	1090	1890	603	194
195	5370	1090	1870	591	199
198	5490	1060	1870	571	202
200	5500	1060	1860	569	204
202	5530	1060	1840	552	206
205	5590	1050	1840	548	209
210	5730	1030	1830	532	214

Additionally, the fission-to-capture ratios in the system analyzed here were only a little smaller for the higher values of Q_{fis} than the lower ones, so the number of captures that take place are also proportional to the number of fissions. When fewer fissions were required (i.e., higher value of Q_{fis}), fewer absorptions occurred in U-235, and less U-236 was produced. Similarly, less Pu-239 and Pu-240 was produced because the number of absorptions in U-238 was also proportional. Thus, the concentrations of U-236, Pu-239, and Pu-240 decreased as the value of Q_{fis} increased (meeting measured results for Pu-239 and Pu-240). However, lower values of Q_{fis} produced more comparable results for U-235 and U-236. Thus, the user should probably use the accepted value of 200 MeV although he/she can enter higher or lower values to tailor the results for specific isotopes.

4.2.2 System-Dependent Changes

One of the largest factors that contributes to errors in *monteburns* is the geometry and material compositions modeled in the system. Although it is primarily up to the user to model the system correctly, a few suggestions are presented here. In particular, the factors discussed in the section are: modeling a system as accurately as possible, using

temperature- and material-dependent factors, and applying appropriate axial boundary conditions.

4.2.2.1 Modeling a System

First, in modeling most reactor systems, it is difficult to include all the details that keep the system at steady-state throughout its life (i.e., keeping track of each rod individually, adding fresh fuel, rotating fuel from one region to another, adjusting the position of the control rods as a function of burnup, changing the soluble boron concentration, etc.). To avoid such complications, computer models commonly combine rods/assemblies into lumped regions, make control rods stationary, and use an average boron concentration in the moderator throughout each burn step. Modeling a larger representative system (i.e., an infinite lattice of assemblies) produces better results than modeling a smaller system (i.e., an infinite number of fuel pins together) because it can take more system-dependent effects into account (i.e., burnable poison fuel rods, control rods, instrumentation tubes, etc.) and more easily keep the model at steady-state. This difference was seen in Test Cases #2 and 3, where both a pin and an assembly case were presented. Because the compositions of surrounding fuel pins in Test Case #2 were not known, it was not possible to model the case as accurately as an assembly to get better results (although neither model would account for leakage or other system-dependent effects). However, it was possible to adjust the amount of soluble boron in the water surrounding the pins to produce a representative spectrum of a critical system (excluding leakage considerations). As can be seen from Table 11, answers were closer to measured values in this system than with the referenced input parameters (although these were used in the test case for a better comparison to the other codes). This is because with a k_{eff} around 1.0, a more realistic spectrum and more representative cross sections were obtained.

Table 11. Results as a Function of K_{eff} and Cross Section (Test Case #2, Scenario A - mg/g UO_2)

Parameters	U-235	U-236	Pu-239	Pu-240
measured value	8.470	3.140	4.264	1.719
k_{eff} from 1.3 to 1.0	8.104	3.206	3.944	1.685
k_{eff} around 1.0 (ENDF/B-V)	8.623	3.178	4.112	1.701
k_{eff} around 1.0 (ENDF/B-VI)	8.463	3.178	4.072	1.681

However, the best spectrum would have been obtained by using a detailed reactor core model, including water surrounding the assembly, the pressure vessel, etc., to account for leakage and other total system effects.

In addition, a comparison of ENDF/B-V and ENDF/B-VI cross sections was performed (see Table 11). The ENDF/B-V libraries produced better results for U-235, but the ENDF/B-VI libraries produced better results for Pu-239 and Pu-240. This is because it has been shown that the neutron flux associated with U-235 in ENDF/B-VI is greater than that in ENDF/B-V in some energy ranges (for example 10^{-4} to 10^{-3} and 0.1 to 1 MeV), while the neutron flux associated with U-238 in those energy ranges is about the same in both ENDF/B-VI and ENDF/B-V.[26] Thus, more U-235 is burned in ENDF/B-VI than ENDF/B-V and less Pu-239 and Pu-240 is created. This reduction in plutonium isotopes could also be a result of the fact that their neutron fluxes in this same energy range in ENDF/B-VI were also higher than those in ENDF/B-V, possibly causing more plutonium atoms to be depleted and matching measured results better. Nonetheless, it is up to the user to determine which data set to use.

In the future it is advisable to model an entire system with as realistic a spectrum as possible to produce the best results in *monteburns*. However, modeling a complex system in MCNP can also significantly increase the run time required, so the user must weigh the benefits of each model against the consequences.

4.2.2.2 Temperature- and Material-Dependent Parameters

Next, the effect of using temperature- and material-dependent parameters in modeling a system is also important. Along with using temperature-dependent cross sections in MCNP (typically processed by NJOY), the temperature of each cell (in MeV) should be included in the MCNP input file using the TMP card.[1] To show this, Test Case #4 with gadolinium in the center pin was run with both temperature-dependent cross sections (xs) and the TMP card in MCNP, temperature-dependent cross sections without this card, and neither. In addition, effect of using S(α,β) treatment for the light water in the system was studied. S(α,β) treatment accounts for the binding effects of hydrogen and oxygen nuclei in light water at thermal energies.[5] This binding affects interactions between thermal neutrons and the material and can be important for LWR systems. The three analyses discussed above used S(α,β) treatment, and the case with temperature-dependent cross sections and the TMP card was rerun without S(α,β) treatment to complete the comparison. The results from these analyses are in Table 12.

Table 12. Effect of Temperature on Power Distribution

Power Fraction	With Temp. Dep. xs and TMP card	With Temp. Dep. xs; no TMP card	No Temp. Dep.	No S(α,β) Treatment	Published Range of Values
Region 1	1.055	1.048	1.063	1.054	1.053 to 1.062
Region 2	0.437	0.485	0.402	0.477	0.413 to 0.460
Region 3	1.086	1.081	1.087	1.077	1.082 to 1.087

As expected, the greatest accuracy was achieved when temperature-dependent cross sections, the TMP card, and S(α,β) treatment were used. In fact, *monteburns* did not even calculate a power distribution in the correct range when temperature-dependent cross sections were included without the TMP card. When neither were included, the

results were close to the published range but were not within it. S(α,β) treatment slightly decreased the accuracy of the results, but not as much as using all temperature parameters for this particular case. Other cases and/or increased/decreased statistics may produce better results or may not make the outcome as exaggerated as it appears here. Nonetheless, it is recommended to include temperature-dependent cross sections, the TMP card, and S(α,β) treatment in the MCNP input files analyzed by *monteburns* to obtain the correct power distribution and other results.

4.2.2.3 Axial Boundary Conditions

Another parameter that can contribute to the accuracy of the results is the axial boundary conditions used in the model. For the models used in all the test cases discussed in Section 4.1, reflective boundary conditions were placed on all six sides of the system being analyzed to simulate that it (i.e., either a pin or an assembly) was one within an infinite lattice of similar ones. These models were consistent with those described in the referenced input in the radial direction, but how the other codes modeled the system in the axial direction was unknown. Because the composition of the material at the ends of the fuel rods in the experimental system was also not specified in the referenced input, it was assumed that all neutrons were reflected back into the rod once they reached the ends (i.e., no leakage occurred). This assumption may not have been fully representative of the experimental reactor because some amount of leakage probably did occur. To quantify this effect, Scenario B of Test Case #2 was rerun with reflective boundary conditions in the axial direction, 10 cm of water on the ends of the each fuel rod, and a vacuum at both ends of each fuel rod (to simulate the maximum amount of leakage). The results of this analysis compared to measured data appear in Table 13.

Table 13. Results of Changes in Axial Parameters (mg/g UO_2)

Isotope	Reflected	Water	Vacuum	Measured
U-235	11.70	11.50	11.60	12.95
U-236	9.08	9.06	9.06	8.84
Np-237	1.02	1.04	1.03	0.89
Pu-238	0.46	0.46	0.46	0.47
Pu-239	10.10	9.99	10.10	10.91
Pu-240	5.47	5.48	5.53	5.61
Pu-241	2.17	2.12	2.14	2.26

This table shows that the differences in the axial representation of the system actually had little effect on the results, although the case with reflective boundary conditions did come the closest to the measured results. This either means that the material at the ends of the fuel rods in the measured system was probably a large scatterer and effectively sent a majority of the neutrons back into the pin, or the pins were long enough that axial edge effects were not important. The amount of leakage that actually occurred was probably slightly larger than that portrayed by reflective boundary conditions and smaller than that with water. Thus, the use of reflective boundary conditions in the axial direction is justified for the test cases in Section 4.1.

Overall, using the best statistics possible without compromising the run time is the key to obtaining the most efficient results. Both by determining optimum input parameters and by modeling the system most effectively, good statistics can be obtained. However, using good statistics often means increasing the required run time of the problem. It is thus up to the user to determine required statistical accuracy and balance this against the run time.

5.0 APPLICATIONS OF MONTEBURNS

Monteburns was written to be applicable for a wide variety of systems, including both reactor and accelerator-driven problems. One of the limitations of other linkage codes between MCNP and ORIGEN2 (discussed in Section 2.3.1) is that they can only be used for relatively simple geometries and may not be applicable for more than one burnup step in an automated fashion (i.e., decay periods following multiple irradiation periods, etc.). *Monteburns* was written to be flexible and accommodating to any type of MCNP input file and irradiation information to minimize limitations, and it is still being modified to incorporate additional options. Two examples of applications for which *monteburns* is currently being used in the Nuclear Design and Analysis Group (TSA-10) at Los Alamos National Laboratory are presented in this section. These are the Accelerator Transmutation of Waste (ATW) project and non-fertile (i.e., non-uranium) fuel applications. Although representative, they are not inclusive of the full spectrum of problems to which *monteburns* can be applied in other groups, laboratories, and industries.

5.1 Accelerator Transmutation of Waste

One of the largest issues currently being addressed in the nuclear industry is what should be done with radioactive waste. Included in this category is spent fuel, which is contained in fuel assemblies removed from nuclear reactor cores after irradiation. This fuel contains significant amounts of plutonium, numerous actinides, and fission products, some of which have relatively long half-lives. The purpose of the ATW project is to design a system to enhance repository performance by reducing long-term radiotoxicity of spent fuel and other high-level wastes by three orders of magnitude (i.e., after processing in ATW, this waste after 300 years should have a lower toxicity than untreated waste after 100,000 years).[3]

For this purpose, the following goals were set for the project:

- Destroy over 99.9% of residual actinides
- Destroy over 99.9% of technetium and iodine (long-lived fission products)
- Separate strontium and cesium (short-lived fission products that significantly contribute to the heat loading of the repository)
- Separate the uranium from the other spent fuel so that it can be stored or re-used and to reduce the amount of plutonium produced during transmutation
- Produce electricity (destruction of actinides could potentially produce energy, which could both power the accelerator and be sold)

The ATW system would be powered by a high-power proton linear accelerator similar to the one being considered for the Accelerator Production of Tritium (APT) project. A pyrochemical spent fuel treatment/waste cleanup system would be used to process the materials remaining after irradiation. The waste itself would be contained in solid waste pins with a configuration similar to the one in Figure 7. The waste transmutation region is designed as three separate zones, where pins in Zone 2 have been irradiated for a cycle in Zone 3, and pins in Zone 1 have been irradiated for one cycle in Zone 3 and one cycle in Zone 2. Once these pins are burned in Zone 1, the material is processed so that the actinides are concentrated to obtain the desired reactivity, and the waste is refabricated into pins and inserted as "fresh" waste into Zone 3. The spallation target would be a heavy metal target made of liquid lead-bismuth eutectic (LBE), which helps produce a high intensity neutron source for the outer zones. The system would operate in a subcritical regime and with a fast neutron spectrum, which allows for more efficient destruction of actinides because the fission-to-capture ratio of many plutonium isotopes and higher actinides is larger at fast energies.

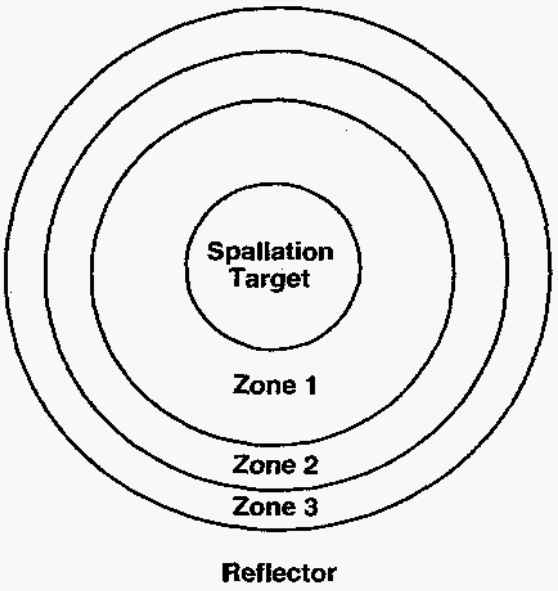


Figure 7. Sample of core configuration for ATW

Monteburns can incorporate all aspects of this design; it moves material from one region to another in MCNP and analyzes the burnup in as many materials as desired for each irradiation step before transferring the resulting material compositions back to MCNP for further analysis. According to preliminary calculations, the following results were both desired and achieved:

- A 2 GW_t ATW can burn almost any combination of higher actinides at a rate of more than 500 kg/yr. with a minimum cycle length of 100 days;
- Technetium can be used as a burnable poison and to harden the spectrum; Tc-99 can be transmuted at a rate greater than 40 kg/yr.; and
- The harder the neutron spectrum, the more efficiently ATW destroys higher actinides because the fission-to-capture ratios of the actinides increase.

Using four-month (121 day) cycles and the feed specified in Table 14, the amount of transmutation/destruction experienced by various actinides in ATW are shown in Table

15. Positive values in this table correspond to production and negative ones to destruction.

Table 14. Feed Material for ATW (kg)

Cycle	U-238	Np-237	Pu-238	Pu-239	Pu-240	Pu-241	Pu-242	Am-241	Tc-99	actinide
1	0	0	0	0	0	0	0	0	0	0
2	100	18	6	207	96	32	19	21	22	502
3	88	16	5	182	84	28	17	18	19	441
4	81	15	5	166	77	25	16	17	18	403
5	76	14	4	157	73	24	15	16	17	382
6	73	13	4	149	69	23	14	15	16	363
7	70	13	4	144	67	22	14	14	15	351
8	68	12	4	141	65	22	13	14	15	342
9	67	12	4	138	64	21	13	14	15	336
10	66	12	4	136	63	21	13	14	15	331
11	66	12	4	135	62	21	13	14	14	328
12	65	12	4	134	62	21	13	13	14	325
13	65	12	4	133	62	20	12	13	14	324
14	64	12	4	132	61	20	12	13	14	322
15	64	12	4	132	61	20	12	13	14	321
16	64	12	4	131	61	20	12	13	14	319
17	64	11	4	131	61	20	12	13	14	318
18	64	11	4	131	60	20	12	13	14	317
19	63	11	4	130	60	20	12	13	14	316
20	63	11	4	130	60	20	12	13	14	316
21	63	11	4	129	60	20	12	13	14	315
22	63	11	4	129	60	20	12	13	14	314
23	63	11	4	129	60	20	12	13	14	313
24	63	11	4	129	59	20	12	13	14	312
25	62	11	3	128	59	20	12	13	14	312
26	62	11	3	128	59	20	12	13	14	311
27	62	11	3	128	59	20	12	13	14	310
28	62	11	3	127	59	20	12	13	14	310
29	62	11	3	127	59	20	12	13	14	309
30	62	11	3	127	59	19	12	13	14	309
31	62	11	3	127	59	19	12	13	14	308
32	62	11	3	126	58	19	12	13	14	307
33	61	11	3	126	58	19	12	13	14	307
34	61	11	3	126	58	19	12	13	14	306
35	61	11	3	126	58	19	12	13	14	306
36	61	11	3	126	58	19	12	13	13	305
37	61	11	3	125	58	19	12	13	13	305
38	61	11	3	125	58	19	12	13	13	304
39	61	11	3	125	58	19	12	13	13	304
40	61	11	3	125	58	19	12	13	13	303
total	2570	461	144	5280	2440	808	493	528	565	12800

Table 15. Amount of Material Produced(+)/Destroyed(-) by ATW (kg)

Cycle	U-238	Np-237	Pu-238	Pu-239	Pu-240	Pu-241	Pu-242	Am-241	Tc-99	actinide
1	-149	-17	10	-214	7	-19	3	-21	-21	-380
2	-156	-16	11	-201	-4	-18	2	-20	-20	-388
3	-159	-16	10	-193	-12	-17	1	-19	-19	-396
4	-100	-16	8	-190	-20	-16	0	-18	-19	-344
5	-88	-15	6	-182	-26	-16	-1	-17	-19	-333
6	-81	-15	4	-173	-32	-16	-1	-16	-18	-323
7	-76	-14	2	-163	-36	-16	-2	-16	-17	-315
8	-73	-14	0	-160	-41	-16	-2	-15	-18	-316
9	-70	-13	-1	-155	-43	-17	-3	-15	-18	-314
10	-68	-13	-2	-152	-47	-17	-4	-15	-18	-313
11	-67	-13	-3	-149	-50	-18	-4	-14	-18	-312
12	-66	-13	-3	-145	-51	-18	-5	-14	-17	-310
13	-66	-12	-3	-141	-52	-18	-5	-14	-17	-307
14	-65	-12	-4	-137	-52	-18	-5	-13	-17	-304
15	-64	-12	-4	-138	-54	-18	-6	-14	-17	-307
16	-64	-12	-4	-137	-55	-19	-6	-13	-17	-308
17	-64	-12	-4	-134	-56	-18	-7	-13	-16	-305
18	-64	-12	-4	-133	-55	-19	-7	-13	-16	-303
19	-64	-12	-4	-132	-57	-19	-7	-13	-16	-303
20	-63	-12	-4	-137	-58	-20	-8	-14	-17	-313
21	-63	-11	-4	-130	-57	-19	-8	-13	-16	-304
22	-63	-12	-4	-131	-58	-19	-8	-13	-16	-306
23	-63	-11	-4	-130	-57	-19	-8	-13	-16	-304
24	-63	-11	-4	-129	-58	-20	-8	-13	-16	-305
25	-62	-11	-4	-130	-57	-20	-9	-13	-15	-305
26	-63	-12	-4	-132	-59	-19	-9	-13	-16	-307
27	-62	-11	-4	-128	-58	-19	-9	-13	-15	-303
28	-62	-11	-4	-129	-58	-19	-9	-13	-16	-305
29	-62	-11	-4	-127	-58	-19	-9	-12	-15	-302
30	-62	-11	-4	-129	-59	-20	-9	-13	-15	-306
31	-62	-11	-4	-128	-58	-19	-10	-13	-15	-304
32	-62	-11	-4	-129	-59	-20	-10	-13	-15	-308
33	-62	-11	-4	-130	-60	-19	-10	-13	-15	-308
34	-61	-11	-4	-127	-59	-20	-11	-13	-15	-305
35	-61	-11	-4	-126	-59	-20	-10	-13	-15	-303
36	-61	-11	-4	-129	-59	-19	-11	-13	-15	-307
37	-61	-11	-3	-124	-58	-19	-11	-12	-14	-301
38	-61	-11	-3	-126	-58	-19	-11	-13	-14	-301
39	-61	-11	-4	-127	-60	-19	-11	-13	-15	-305
40	-61	-11	-4	-127	-60	-19	-11	-13	-15	-306
total	-2906	-495	-64	-5731	-1962	-738	-257	-564	-658	-12587

With an initial system input of about 2300 kg of actinides and 700 kg Tc-99 and a steady-state feed rate of approximately 320 kg of actinides and 14 kg Tc-99 per four-

month cycle, over 900 kg of actinides and around 45 kg of Tc-99 are destroyed per year. This successfully exceeded the goals of 500 kg and 40 kg per year, respectively.

5.2 Plutonium Destruction

Although *monteburns* was initially designed for the ATW project, it has been expanded (and tested as shown in Section 4.1) for reactor uses. One of the current uses of *monteburns* in a reactor-based system is to study various parameters and fuel cycle concepts for their effectiveness in the destruction of plutonium. There is a great deal of reactor-grade plutonium currently contained in spent fuel that may become a proliferation risk in the next century if it is not destroyed. In addition, there are about 50 metric tons of surplus weapons-grade plutonium in the US being proposed for disposition, possibly in a reactor.[27]

Studies are currently being performed to determine the best way of destroying this plutonium, including examining different fuel forms, plutonium isotopic compositions, and neutron energy spectra. Figure 8 shows the percentage of plutonium destroyed in each system as a function of burnup. Unless stated otherwise, the parameters used in this figure were: non-fertile fuel (described below), reactor-grade (RG) plutonium, and a light-water reactor system.

5.2.1 Fuel Form

First, the two fuel forms being investigated are: MOX fuel (*monteburns* calculations for this fuel were demonstrated in Test Case #5), and non-fertile (NF) fuel (plutonium dioxide (PuO_2) in a calcia (CaO)-stabilized zirconium dioxide (ZrO_2) matrix with an erbia (ErO_2) poison).[12] The MOX fuel modeled in this analysis consisted of 93w% depleted uranium oxide and 7w% RG PuO_2, and the non-fertile fuel was comprised of 7w% RG PuO_2, 1w% ErO_2, 85.6w% ZrO_2, and 6.4w% CaO. The purpose of using non-fertile fuel for the destruction of plutonium is to transmute plutonium actinides

without building them. The absence of uranium in the NF fuel leads to a lack of production of plutonium due to transmutation of the uranium isotopes, and hence to higher destruction rates. Thus, from Figure 8, it can be seen that the non-uranium-based NF fuel allows better net plutonium destruction than MOX fuel and should be further considered for this purpose.

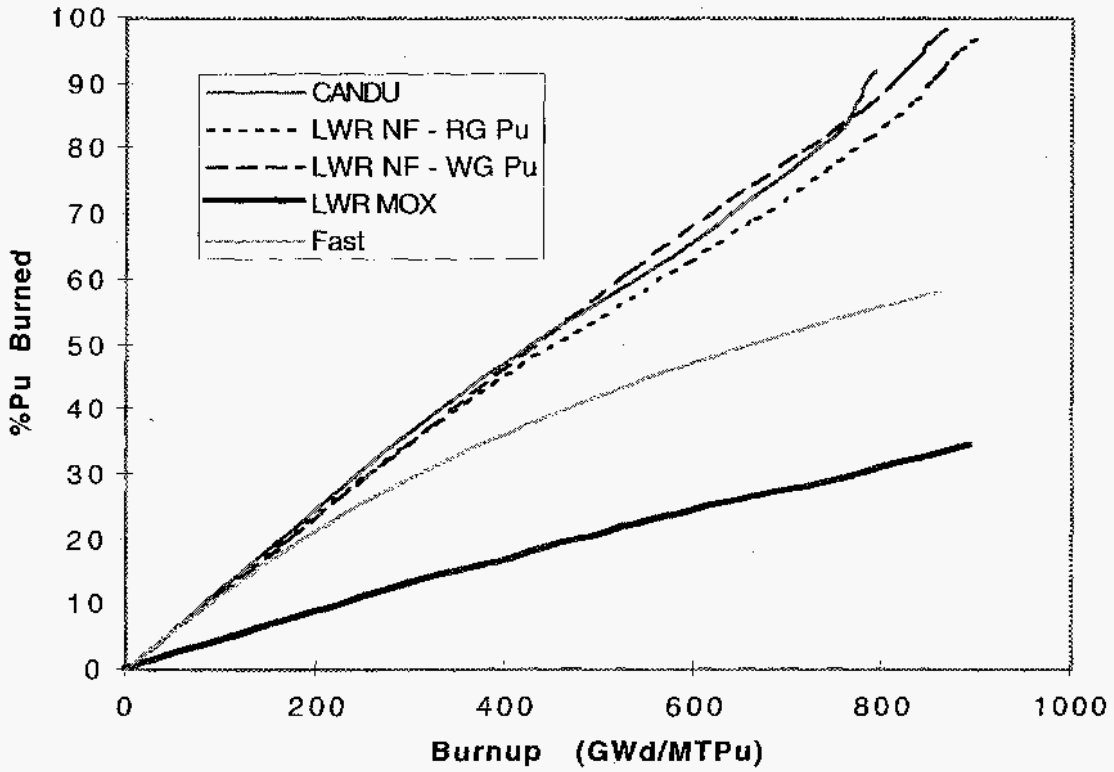


Figure 8. Plutonium Destruction as a Function of Burnup

5.2.2 Isotopic Composition

Second, the initial plutonium isotopes in the fuel also influence how effectively plutonium is destroyed. This is because the fission-to-capture ratio of every plutonium isotope is different, and the higher this value is, the more fissions occur relative to transmutations, and the more plutonium is destroyed (instead of higher actinides built

up). The two plutonium isotopes with the largest fission-to-capture ratios are Pu-239 and Pu-241 (see Table 4a for sample values) because they are fissile isotopes. Thus, the more Pu-239 and Pu-241 that exist in the plutonium relative to other plutonium isotopes (such as Pu-238 and Pu-240), the faster the plutonium fissions and is destroyed. Some plutonium can also be destroyed through decay of Pu-241 to americium, but not as fast as that which fissions. However, specifying the composition of the plutonium isotopes in the material is not an option, so although this is not an input parameter, it is shown here solely for comparison purposes. The two types of plutonium compared in this example were reactor-grade plutonium (with a representative composition of 1.57w% Pu-238, 57.54w% Pu-239, 26.65w% Pu-240, 8.85w% Pu-241, and 5.39w% Pu-242), and weapons-grade plutonium (with an average composition of 93w% Pu-239 and 7w% Pu-240).

As expected, the weapons-grade plutonium was destroyed faster than the reactor-grade because it initially contained more fissile Pu-239 atoms than non-fissile Pu-240 ones. Pu-240 is more likely to transmute than fission, so a material starting with more Pu-240 has only one main chance to fission (when it is Pu-241) before it transmutes to higher actinides whereas Pu-239 atoms have two main chances (Pu-239 and Pu-241). The number of fissions that take place in the system must be the same in both cases, so higher actinides are probably contributing to relatively more fission interactions in the former case than in the latter case, which is why less net destruction of plutonium occurs.

5.2.3 Energy Spectrum

Finally, the energy spectrum of neutrons in the system in which the fuel is being irradiated also contributes to the results. The three different spectra analyzed in this example were a representative light-water, heavy-water, and fast system. The first two of these systems were modeled in *monteburns* as one assembly of NF fuel surrounded by a matrix of system-representative fuel assemblies (i.e., LEU fuel in a PWR[28] for the

LWR case and depleted-uranium CANDU assemblies[29] for the heavy water case) to keep the k_{eff} of the system around 1.0. The third, a fast system, was difficult to model in MCNP without a detailed system design for this purpose, so an ORIGEN2 run using cross sections representative of the Fast Flux Test Facility (FFTF) was performed instead.

By comparing the LWR RG Pu case to the CANDU and Fast cases run with RG Pu, Figure 8 indicates that the heavy-water (CANDU) system was the most effective in destroying plutonium, which is probably a result of the fact that fission-to-capture ratios were greater for it than for the light-water system (see Table 16). This is because a heavy-water system has a more thermal spectrum than a LWR, and neutrons probably avoid many of the absorption resonances. In addition, neutrons can be absorbed in hydrogen at thermal energies in a LWR system, whereas they are absorbed and/or fission in plutonium isotopes instead in the heavy-water system.

Table 16. Fission-to-Capture Ratios of Isotopes in Each Spectrum

Isotope	Light-Water	Heavy-Water	Fast
U-235	3.4 to 5.3	4.46 to 5.64	3.8
Pu-239	1.78 to 1.88	~1.98	4.59
Pu-241	2.77 to 2.75	2.91 to 2.78	6.02

Table 16 also indicates that the fission-to-capture ratios for the plutonium isotopes in the fast system were also relatively large, which means that the neutron energies were large enough that they avoided resonances altogether and primarily fissioned instead. Thus, plutonium should have been destroyed more quickly with this fast system than the thermal ones, but Figure 8 shows that this is not the case at high burnups. This is probably because the fast system was modeled in ORIGEN2 instead of *monteburns*, and system-dependent effects were not taken into account as a function of burnup. The

results from this example shows the importance of using *monteburns* instead of just ORIGEN2. Nonetheless, a LWR is the most probable system that would be used for the destruction of plutonium because some are already operating in the US Even though a heavy water system may produce slightly better results, there are a number of political hurdles that must be addressed before a heavy-water reactor is built in the US or the Canadian CANDU reactors can be used.

In conclusion, the reason that *monteburns* is ideal for this type of analysis is that it models any type of system accurately and provides spectrum-dependent fluxes and cross sections for a system at each irradiation step. As such, the effect of each parameter varied in this example influences the results in the most realistic computer model possible. In particular, for a system with plutonium in the form of NF fuel, the power produced in the fuel decreases significantly over time (thus other types of material besides NF fuel must be present in a reactor to keep it critical), and the fuel reaches fairly large burnups. The flux distribution as well as the cross sections change significantly with this power loss. It is important to use a code such as *monteburns* that can account for such changes.

6.0 LIMITATIONS OF AND FUTURE WORK FOR *MONTEBURNS*

One limitation of *monteburns* is that it is currently designed to run only on a UNIX system. Not all users may have this type of system, and *monteburns* is not yet capable of running on VMS or PC systems. Significant changes must be made to the command file (currently a c-shell file), and minor modifications must be made to the FORTRAN77 file so that the code can operate on any type of machine and/or system.

Monteburns currently only extracts a few reactor physics constants (η, υ, etc.) from MCNP output files. It can, however, be modified in the future to extract more values, depending on what uses the program may eventually have. It may also be modified to calculate activation and decay powers, and the input may be simplified further to make it even more user-friendly. Any of these suggestions should enhance the capability and versatility of the code.

Another modification that could be made to *monteburns* is to allow it to interface with another burnup code besides ORIGEN2. Examples of such codes include ORIGEN-S (part of the SCALE package) and CINDER90 (primarily used for calculations involving accelerator-driven systems).[17] Whether the benefit of this addition is great enough to offset the additional requirement of more complex input has yet to be determined. All of these limitations can be resolved by modifying the FORTRAN77 program and/or the c-shell executable.

Throughout this document, references to resonance self-shielding and the variable increase or decrease of cross sections with burnup are mentioned. However, no detailed analyses were performed to determine how resonances affect the value of the flux or the effective cross sections. A detailed analysis could be performed in the future to study these affects and determine exactly why the results presented in this document were obtained. This, along with the activities discussed above, constitutes the proposed future work activities for *monteburns*.

7.0 CONCLUSIONS

This document provided a thorough description and benchmarking results of the automated burnup code *monteburns*, which links the transport code MCNP and the radioactive decay and burnup code ORIGEN2. This linkage code was designed to limit the amount of information the user is required to input and still perform detailed, automated burnup calculations for any type of system and number of irradiation periods. The advantages it has over other burnup codes are: 1) it allows the user to model a detailed, 3-D system, 2) it modifies material cross sections as a function of burnup and flux distributions within a system, 3) it offers a variety of options and allows system changes to be made frequently throughout a burn interval, and 4) it is fully automated and relatively easy to learn. The purpose of this document is not only to serve as a thesis but is also to assist those who plan to use *monteburns* by providing a validation of the code and discussions of "tricks" found useful when running the code.

Monteburns is comprised of a combination of a c-shell UNIX executable file and a FORTRAN77 program and primarily acts as a pre- and post-processor for ORIGEN2 and a post-processor for MCNP. The main calculations that it performs are: 1) the recoverable energy per fission according to the distribution of actinides in the system, and 2) the conversion of the flux calculated by MCNP for a region(s) to the actual flux seen by that region as well as the power produced by the region. Only two main input files are required for *monteburns* (others are optional): 1) a working MCNP input file, and 2) a *monteburns* input file containing a list of parameters relevant to the system being analyzed. A number of variables are currently output, including reactor physics constants, cross sections, and compositions of materials in the system before and after each step. The code is frequently being updated and modified to suit user's needs and desires.

The most important portion of this document is the benchmarking section, which showed that *monteburns* performs burnup calculations just as well as or better than those

performed using other codes. Different geometries, fuel types, and reactor systems were modeled and compared to measured and/or published calculations from other codes, and the errors/differences obtained by these comparisons were all considered to be acceptable. In addition, a number of statistical analyses were performed for *monteburns*, both to analyze the effect(s) of several input parameters on the results and to describe the importance of modeling the system as accurately a fashion as possible. Some examples of problems for which *monteburns* is currently being used were presented as well, along with suggestions of future work that may be performed for *monteburns*.

In conclusion, the code *monteburns* has now been described and benchmarked for the burnup scenarios in Section 4.1. It produces comparable results to other well-known burnup codes, such as those in the SCALE suite of programs. *Monteburns* is a straightforward yet versatile solution requiring little training other than that required for MCNP and will soon be publicly available through the Radiation Safety Information Computational Center (RSICC) at Oak Ridge National Laboratory.

APPENDICES

APPENDIX A. LISTING OF C-SHELL FILE *MONTEBURNS*

```
#!/bin/csh
#  Version 4  September 1998
date
cp $1.inp mb.inp
#
# File management ---------------------------------
if (-e $1.feed) cp $1.feed feed
if (-e tmpfile) then
else
mkdir tmpfile
endif
#
# Get shell variables -----------------------------
monteb a
@ nout = `awk '$2 == "nout" {print int($1)}' ./tmpfile/params`
@ npre = `awk '$2 == "npre" {print int($1)}' ./tmpfile/params`
@ nrst = `awk '$2 == "nrst" {print int($1)}' ./tmpfile/params`
@ nkeff = `awk '$2 == "nkeff" {print int($1)}' ./tmpfile/params`
@ nmat = `awk '$2 == "nmat" {print int($1)}' ./tmpfile/params`
echo $nout  $npre  $nrst  $nkeff  $nmat
#
echo ...MonteBurns: Write natural element and origen input files
monteb e
monteb 5
#
if ($nrst == 0) then
# Set up initial run ------------------------------
# ..Backup fort.9
#
@ i3 = 1
while ($i3 <= $nmat)
if (-e fort.9.0) then
cp fort.9.0 fort_$i3.9
else
cp fort_$i3.9 fort.9.0
endif
@ i3 ++
end
echo ...MonteBurns: Delete Old MCNP Files
if (-e mbmcm ) rm mbmcm
if (-e mbmco ) rm mbmco
if (-e mbmcr ) rm mbmcr
if (-e mbmcs ) rm mbmcs
echo ...MonteBurns: Check Print Card and create skeleton mcnp input
monteb 1 <$1
echo ...MonteBurns: Run MCNP Input Module to get initial comps
mcnp ix n=mbmc
echo ...MonteBurns: Write tally file tal2.inp
# Get number of predictors from status
```

```
monteb 2 <mbmco
echo ...MonteBurns: Write initial origen comp file fort.7 and nat isos
monteb 4 <mbmco
@ i1 = 0
else
#
# Set up restart run ------------------------
@ i1 = $nrst + 1
@ i3 = 1
while ($i3 <= $nmat)
cp ./tmpfile/fort9_$i3.$nrst fort_$i3.9
cp ./tmpfile/fort7_$i3.$nrst fort_$i3.7
if (-e ./tmpfile/mbori_$i3.$i1.tmp) then
else
cp ./tmpfile/mbori_$i3.$i1 ./tmpfile/mbori_$i3.$i1.tmp
endif
@ i3 ++
end
cp ./tmpfile/mbmc.$nrst mbmc
cp ./tmpfile/mbinp.$nrst mb.inp
endif
#
# Beginning of outer loop ----------------------
while ($i1 <= $nout)
#
echo ...MonteBurns: Begin outer loop $i1
#
# tally nrst in mb.inp so monteb knows what step
#
if ($i1 > 0) monteb 9
#
# determine material in each MCNP region
#
if ($i1 > 0) monteb c
@ i3 = 1
while ($i3 <= $nmat)
if ($i1 > 0 ) then
mv ./tmpfile/mbori_$i3.$i1.tmp ./tmpfile/mbori_$i3.$i1
cp ./tmpfile/mbori_$i3.$i1 mbori_$i3
@ nval = `awk '$2 == "nval" {print int($1)}' ./tmpfile/param3_$i3`
#
# see if the same material is present in each region and if not,
# copy new material to current $i3 value fort.7 file
#
if ($nval == 0) then
cp fort_$i3.7 fort_$i3.7.tmp
cp mnat_$i3.tmp mnat_$i3.t.tmp
else
if ($nval != $i3) then
cp fort_$nval.7 fort_$i3.7.tmp
cp mnat_$nval.tmp mnat_$i3.t.tmp
```

```
else
cp fort_$i3.7 fort_$i3.7.tmp
cp mnat_$i3.tmp mnat_$i3.t.tmp
endif
endif
endif
@ i3 ++
end
@ i3 = 1
while ($i3 <= $nmat)
if ($i1 > 0) then
mv fort_$i3.7.tmp fort_$i3.7
mv mnat_$i3.t.tmp mnat_$i3.tmp
endif
cp fort_$i3.7 fort_$i3.4
@ i3 ++
end
#
if ($i1 == 1) then
@ npre2 = $npre + 1
else
@ npre2 = $npre
endif
if ($i1 == 0) @ npre2 = 1
#
@ i2 = 1
@ ndsc = 0
@ i3 = 1
while ($i3 <= $nmat)
if (-e ./tmpfile/param_$i3.$i1) then
@ ndisc = `awk '$2 == "ndisc" {print int($1)}' ./tmpfile/param_$i3.$i1`
if ($ndisc == 1) then
@ ndsc = 1
endif
endif
@ i3 ++
end
if ($ndsc == 1) then
echo ...Monteburns: Add discrete feed to fort.7
monteb b
@ i3 = 1
while ($i3 <= $nmat)
mv fort_$i3.7.tmp fort_$i3.7
cp fort_$i3.7 fort_$i3.4
@ i3 ++
end
if ($nkeff == 1) then
echo ...Monteburns: Add discrete feed to mcnp input file
monteb 7b
cp mbmc.tmp mbmc.sk1
cp mbmc.sk1 mbmc.temp
```

```
@ i3 = 1
while ($i3 <= $nmat)
cat mb7t_$i3.out mb7_$i3.out > mb7t_$i3.tmp
mv mb7t_$i3.tmp mb7t_$i3.out
cat mbmc.temp mat_$i3.inp > mbm.tmp
mv mbm.tmp mbmc.temp
@ i3 ++
end
mv mbmc.temp mbmc
echo ...MonteBurns:  Run MCNP for discrete feed
if (-e mbmcm) rm mbmcm
if (-e mbmco) rm mbmco
if (-e mbmcr) rm mbmcr
if (-e mbmcs) rm mbmcs
mcnp n=mbmc
monteb 6b
cat mb11t.out mb11.out > mb11t.tmp
mv mb11t.tmp mb11t.out
cat mb13t.out mb11.out > mb13t.tmp
mv mb13t.tmp mb13t.out
endif
endif
#
# Determine grams of feed at the beginning of each step
monteb 8b
@ i3 = 1
while ($i3 <= $nmat)
cat mb12t_$i3.out mb12_$i3.out > mb12t_$i3.tmp
cat mb12a_$i3.out mb12x_$i3.out > mb12a_$i3.tmp
mv mb12t_$i3.tmp mb12t_$i3.out
mv mb12a_$i3.tmp mb12a_$i3.out
@ i3 ++
end
#
# Begninning of inner loop ----------------------
while ($i2 <= $npre2)
if ($i1 > 0) then
echo ...MonteBurns: Run origen predictor $i2 for outer $i1
@ i3 = 1
while ($i3 <= $nmat)
cp mbori_$i3 mbori
cp fort_$i3.9 fort.9
cp fort_$i3.4 fort.4
origen2 <mbori >mboro
mv fort.9 fort_$i3.9
mv fort.7 fort_$i3.7
@ i3 ++
end
endif
echo ...Monteburns: Determine important players / make new mcnp mat
monteb 7m
```

```
cp mbmc.tmp mbmc.skl
echo ...MonteBurns: Write new mcnp tallies and cat new mcnp input
monteb 3
echo ...MonteBurns: Create complete MCNP input file
cp mbmc.skl mbmc
@ i3 = 1
while ($i3 <= $nmat)
cat mb7t_$i3.out mb7_$i3.out > mb7t_$i3.tmp
mv mb7t_$i3.tmp mb7t_$i3.out
cat mbmc mat_$i3.inp tal1_$i3.inp tal2_$i3.inp tal3_$i3.inp > mbmc.temp
mv mbmc.temp mbmc
rm tal1_$i3.inp tal3_$i3.inp
@ i3 ++
end
echo ...MonteBurns: Run MCNP
if (-e mbmcm ) rm mbmcm
if (-e mbmco ) rm mbmco
if (-e mbmcr ) rm mbmcr
if (-e mbmcs ) rm mbmcs
mcnp n=mbmc
echo ...MonteBurns: Modify orig xs file fort.9 and mbori with new flux
monteb 6m
@ i3 = 1
while ($i3 <= $nmat)
if ($i1 > 0 ) mv mbori_$i3.tmp mbori_$i3
mv fort_$i3.9.tmp fort_$i3.9
cat mb4a_$i3.out mb6_$i3.out > mb4a_$i3.tmp
mv mb4a_$i3.tmp mb4a_$i3.out
@ i3 ++
end
@ i2 ++
end
cat mb11t.out mb11.out > mb11t.tmp
mv mb11t.tmp mb11t.out
if ($i1 == 0) then
if ($nkeff == 1) then
cat mb13t.out mb11.out > mb13t.tmp
mv mb13t.tmp mb13t.out
endif
endif
# End of inner loop ------------------------------
#
if ($i1 > 0) then
echo ...MonteBurns: Run origen to compare 1/2 way comps
@ i3 = 1
while ($i3 <= $nmat)
cp fort_$i3.9 fort.9
cp fort_$i3.4 fort.4
cp mbori_$i3 mbori
origen2 <mbori >mboro
mv fort.7 fort_$i3.7
```

```
mv fort.9 fort_$i3.9
@ i3 ++
end
#
monteb 8e
@ i3 = 1
while ($i3 <= $nmat)
cat mb4b_$i3.out mb5_$i3.out > mb4b_$i3.tmp
mv mb4b_$i3.tmp mb4b_$i3.out
@ i3 ++
end
#
# Remove 1/2 way predictor stuff in mbori
monteb 0
@ i3 = 1
while ($i3 <= $nmat)
mv mbori_$i3.tmp mbori_$i3
echo ...MonteBurns: Run origen for complete outer step $i1
cp fort_$i3.9 fort.9
cp fort_$i3.4 fort.4
cp mbori_$i3 mbori
origen2 <mbori >mboro
mv fort.7 fort_$i3.7
mv fort.9 fort_$i3.9
cp fort_$i3.9 ./tmpfile/fort9_$i3.$i1
@ i3 ++
end
#
#  Save stuff for restart -------------------
#
cp mbmc ./tmpfile/mbmc.$i1
cp mb.inp ./tmpfile/mbinp.$i1
#
#  Calculate k-eff at end of burn step ------
#
if ($nkeff == 1) then
echo ...MonteBurns: Determine important players / make new mcnp mat
monteb 7e
cp mbmc.tmp mbmc.skl
cp mbmc.skl mbmc.temp
@ i3 = 1
while ($i3 <= $nmat)
cat mb7t_$i3.out mb7_$i3.out > mb7t_$i3.tmp
mv mb7t_$i3.tmp mb7t_$i3.out
cat mbmc.temp mat_$i3.inp > mbm.tmp
mv mbm.tmp mbmc.temp
@ i3 ++
end
mv mbmc.temp mbmc
echo ...MonteBurns: Run MCNP for complete outer step $i1
if (-e mbmcm ) rm mbmcm
```

```
if (-e mbmco ) rm mbmco
if (-e mbmcr ) rm mbmcr
if (-e mbmcs ) rm mbmcs
mcnp n=mbmc
monteb 6e
cat mb11t.out mb11.out > mb11t.tmp
mv mb11t.tmp mb11t.out
endif
#
#   Remove discrete removal group elements
#
monteb d
@ i3 = 1
while ($i3 <= $nmat)
if (-e fort_$i3.7.tem) mv fort_$i3.7.tem fort_$i3.7
cp fort_$i3.7 ./tmpfile/fort7_$i3.$i1
@ i3 ++
end
endif
#
monteb 8e
@ i3 = 1
while ($i3 <= $nmat)
cat mb5t_$i3.out mb5_$i3.out > mb5t_$i3.tmp
cat mb5tx_$i3.out mb5x_$i3.out > mb5tx_$i3.tmp
mv mb5t_$i3.tmp mb5t_$i3.out
mv mb5tx_$i3.tmp mb5tx_$i3.out
@ i3 ++
end
if ($i1 > 0) then
monteb z
@ i3 = 1
while ($i3 <= $nmat)
cat mb9t_$i3.out mb9_$i3.out > mb9t_$i3.tmp
mv mb9t_$i3.tmp mb9t_$i3.out
@ i3 ++
end
endif
#
# Copy to output files ----------------------
@ i3 = 1
while ($i3 <= $nmat)
cat mb1t_$i3.out mb1_$i3.out > mb1t.tmp
mv mb1t.tmp mb1t_$i3.out
cat mb6t_$i3.out mb6_$i3.out > mb6t.tmp
mv mb6t.tmp mb6t_$i3.out
cat mb2t_$i3.out mb2_$i3.out > mb2t.tmp
mv mb2t.tmp mb2t_$i3.out
cat mb3t_$i3.out mb3_$i3.out > mb3t.tmp
mv mb3t.tmp mb3t_$i3.out
cat mb8t_$i3.out mb8_$i3.out > mb8t.tmp
```

```
mv mb8t.tmp mb8t_$i3.out
cat mb4b_$i3.out mb4_$i3.out > mb4b.tmp
mv mb4b.tmp mb4b_$i3.out
@ i3 ++
end
#
@ i1 ++
echo $nout $i1
end
# End of outer loop ----------------------------------
@ i3 = 1
while ($i3 <= $nmat)
cat mb1 mb1t_$i3.out > mb1.tmp
mv mb1.tmp mb1
cat mb2 mb2t_$i3.out > mb2.tmp
mv mb2.tmp mb2
cat mb3 mb3t_$i3.out > mb3.tmp
mv mb3.tmp mb3
cat mb4a mb4a_$i3.out > mb4a.tmp
mv mb4a.tmp mb4a
cat mb4b mb4b_$i3.out > mb4b.tmp
mv mb4b.tmp mb4b
cat mb5 mb5t_$i3.out > mb5.tmp
mv mb5.tmp mb5
cat mb6 mb6t_$i3.out > mb6.tmp
mv mb6.tmp mb6
cat mb7 mb7t_$i3.out > mb7.tmp
mv mb7.tmp mb7
cat mb8 mb8t_$i3.out > mb8.tmp
mv mb8.tmp mb8
cat mb9 mb9t_$i3.out > mb9.tmp
mv mb9.tmp mb9
cat mb10 mb10t_$i3.out > mb10.tmp
mv mb10.tmp mb10
cat mb12 mb12t_$i3.out > mb12.tmp
mv mb12.tmp mb12
@ i3 ++
end
if ($nkeff == 1) then
cat mb11t.out mb13t.out > crit
else
cp mb11t.out crit
endif
cat crit mb1 mb6 mb2 mb3 mb8 mb12 mb5 mb9 mb10 > $1.mbtmp
cat mb4a mb4b mb7 > $1.mbchk
if (-e feed) then
cat $1.mbtmp feed > $1.mbout
else
mv $1.mbtmp $1.mbout
endif
#
```

```
txt2ps-sw $1.mbout > $1.ps
txt2ps-sw $1.mbchk > $1c.ps
txt2ps-xs $1.mbout > $1.pss
txt2ps-xs $1.mbchk > $1c.pss
#
echo ...Monteburns: Completed
date
```

APPENDIX B. LISTING OF FORTRAN77 PROGRAM *MONTEBF*

```
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c   Version 4  September 1998
c   For info please contact Dave Poston (505)-667-4336 - poston@lanl.gov
c   or  Holly Trellue (505)-665-9539 - trellue@lanl.gov
c
c...MONTEB call a variety of subroutines based on call line ARG
c
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
c
c...Read in command line argument getarg for sun, igetarg for HP
c
      character arg*8
      call getarg(1,arg)
c      call igetarg(1,arg,1)
c
c...Read in input file into common block data from standard input
(initial
c... read (with arg = a) is different than preceding ones b/c nauto
c... has not yet been defined.
c
      if (arg.eq.'a'.or.arg.eq.'wparams') then
        call read
      else
        call readco
      endif
c
c...execute based on arg
c
      if (arg.eq.'1'.or.arg.eq.'pcard')    call pcard
      if (arg.eq.'2'.or.arg.eq.'wtally2')  call wtal2
      if (arg.eq.'3'.or.arg.eq.'wtally')   call wtally
      if (arg.eq.'4'.or.arg.eq.'worcomp')  call worcom
      if (arg.eq.'5'.or.arg.eq.'worinp')   call worinp
      if (arg.eq.'6b'.or.arg.eq.'worxsb')  then
         posit = 'b'
         call worxs
      elseif (arg.eq.'6m'.or.arg.eq.'worxsm') then
         posit = 'm'
         call worxs
      elseif (arg.eq.'6e'.or.arg.eq.'worxse') then
         posit = 'e'
         call worxs
      endif
      if (arg.eq.'7b'.or.arg.eq.'wmcinpb') then
         posit = 'b'
```

```
         call wmcinp
      elseif (arg.eq.'7m'.or.arg.eq.'wmcinpm') then
         posit = 'm'
         call wmcinp
      elseif (arg.eq.'7e'.or.arg.eq.'wmcinpe') then
         posit = 'e'
         call wmcinp
      endif
      if (arg.eq.'8b'.or.arg.eq.'gramsb') then
         posit = 'b'
         call grams
      elseif (arg.eq.'8e'.or.arg.eq.'gramse') then
         posit = 'e'
         call grams
      endif

      if (arg.eq.'9'.or.arg.eq.'wmbinp') then
        nrst=nrst+1
        call wmbinp
      end if
      if (arg.eq.'0'.or.arg.eq.'rmhalf')   call rmhalf(nmat)
      if (arg.eq.'b'.or.arg.eq.'discrete') call discr
      if (arg.eq.'c'.or.arg.eq.'region')   call region
      if (arg.eq.'d'.or.arg.eq.'discremo') call dremo
      if (arg.eq.'e'.or.arg.eq.'natelem')  call natele
      if (arg.eq.'z'.or.arg.eq.'burncalc') call burnca
c
c...Write variables 'params' to be read by shell and make more detailed
c... user's input file
c
      if (arg.eq.'a'.or.arg.eq.'wparams')  then
         call wparam
         call wmbinp
      endif
c
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...WPARAMS writes scratch file containing variables to be read by
c...shell with the AWK command
c
      subroutine wparam
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
c
      open (11,file='./tmpfile/params',status='unknown')
      write (11,902) nouter
      write (11,903) npre
      write (11,904) nrst
```

```
      write (11,905) nkeff
      write (11,906) nmat
      close (11)
c
  902 format (i4,'  nout')
  903 format (i4,'  npre')
  904 format (i4,'  nrst')
  905 format (i4,'  nkeff')
  906 format (i4,'  nmat')
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...READCOM reads in common block data from input file
c
      subroutine read
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      character nisor5*5,met*1
c
c...Read mburn input file, read twice to get niso & nisn, have to
c...read as real variable first and then convert to integer so that it
c...works both on Sun and HP.
c
      open (11,file='mb.inp',status='old')
      read (11,'(a72)') title
      read (11,*) nmat
      do 20 j=1,nmat
   20 read (11,*) mt(j)
      do 30 j=1,nmat
   30 read (11,*) voli(j)
      read (11,*) pow
      read (11,*) qu235
      read (11,*) days
      read (11,*) nouter
      read (11,*) ninner
      read (11,*) npre
      read (11,*) nrst
      read (11,'(a2)') olib
      read (11,'(a72)') locale
      read (11,*) frimp
      read (11,*) nkeff
      do 60 j=1,nmat
      read (11,*) nauto(j)
      ntot(j) = nauto(j)
      do 60 i=1,ntot(j)
      read (11,'(a10)') niso(i,j)
```

```
      backspace (11)
      read (11,'(f6.1)') x
   60 nisn(i,j)=x
      close (11)
c
c...Assign origin iso names
c
      do 10 j=1,nmat
      do 10 i=1,ntot(j)
      nisor5=niso(i,j)
      met='0'
      if (nisor5.eq.'95242') met='1'
   10 nisor(i,j)=nisor5//met
c
      do 15 j=1,nmat
      do 15 i=1,ntot(j)
      nisnr(i,j)=nisn(i,j)*10
      if (nisnr(i,j).eq.952420) nisnr(i,j)=nisnr(i,j)+1
   15 continue
c
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...READCOM reads in common block data from input file
c
      subroutine readco
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      character nisor5*5,met*1
c
c...Read mburn input file, read twice to get niso & nisn, have to
c...read as real variable first and then convert to integer so that it
c...works both on Sun and HP.
c
      open (11,file='mb.inp',status='old')
      read (11,'(a72)') title
      read (11,*) nmat
      do 20 j=1,nmat
   20 read (11,*) mt(j)
      do 30 j=1,nmat
   30 read (11,*) voli(j)
      read (11,*) pow
      read (11,*) qu235
      read (11,*) days
      read (11,*) nouter
      read (11,*) ninner
```

```
      read (11,*) npre
      read (11,*) nrst
      read (11,'(a2)') olib
      read (11,'(a72)') locale
      read (11,*) frimp
      read (11,*) nkeff
      do 60 j=1,nmat
      read (11,*) nauto(j)
      read (11,*) ntot(j)
      do 60 i=1,ntot(j)
      read (11,'(a10)') niso(i,j)
      backspace (11)
      read (11,'(f6.1)') x
   60 nisn(i,j)=x
      close (11)
c
c...Assign origin iso names
c
      do 10 j=1,nmat
      do 10 i=1,ntot(j)
      nisor5=niso(i,j)
      met='0'
      if (nisor5.eq.'95242') met='1'
   10 nisor(i,j)=nisor5//met
c
      do 15 j=1,nmat
      do 15 i=1,ntot(j)
      nisnr(i,j)=nisn(i,j)*10
      if (nisnr(i,j).eq.952420) nisnr(i,j)=nisnr(i,j)+1
   15 continue
c
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...PCARD checks mcnp input file for print card, and alters or adds one
c...(only run once at beginning of monteburns)
c
      subroutine pcard
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      character ju5*5,ju80*80,m(20)*1,file1t*12,file2t*12
      character file3t*12,file4a*12,file4b*12,file5t*12,file5x*12
      character file7t*12,file8t*12,fname*12,fil12t*12,fil12a*12
c
      open (12,file='mbmc',status='unknown')
c
```

```
   10 k=k+1
      read (5,'(a5)',end=15) ju5
      if (ju5.eq.'print') ni=k
      goto 10
   15 rewind (5)
   20 do 30 n=1,k-1
      read (5,'(a80)') ju80
      if (n.ne.ni) then
        write (12,'(a80)') ju80
      else
        write (12,'(a8)') 'print 40'
      end if
   30 continue
      if (ni.eq.0) write (12,'(a8)') 'print 40'
c
      close (12)
c
c...Remove mt card and write mbmc.skl
c
      open (11,file='mbmc',status='old')
      open (12,file='mbmc.skl',status='unknown')
c
      iflag=0
      n = 0
   40 read (11,'(20a1)',end=50) (m(i),i=1,20)
      ifd = 0
      nogo=0
      do 45 i=1,20
   45 if (m(i).ne.' ') nogo=1
c
c...Determine numerical value of material
c
      if (nogo.eq.0.and.iflag.eq.1) goto 40
      if (m(1).eq.'m') then
        iflag = 1
        do 47 i=6,2,-1
         if (m(i).eq.' ') ii=i
   47   continue
         matr=0
        do 48 i=2,ii-1
   48   matr=matr+(ichar(m(i))-48)*10**(ii-1-i)
c
c... Identify if MCNP material is one of the user requested materials
c
do 49 j=1,nmat
   49 if (matr.eq.abs(mt(j))) ifd=1
      end if
c
c...  Print  lines  excluding  user-specified  material  identifiers  to
skeleton
c
```

```
      if (ifd.eq.0) then
        backspace(11)
        read (11,'(a80)') ju80
        write (12,'(a80)') ju80
        goto 40
      else
c
c... If MCNP material is equal to user specified one, then print
material
c... identification cards to appropriate output file.   Remove blank
lines
c... from end of MCNP input file
c
        do 52 j=1,nmat
        if (matr.eq.abs(mt(j))) then
          if (j.lt.10) then
            fname = 'mat_'//char(j+48)//'.inp'
          elseif (j.ge.10) then
            j1 = j/10
            j2 = j - j1*10
            fname = 'mat_'//char(j1+48)//char(j2+48)//'.inp'
          endif
          open (13,file=fname,status='unknown')
          n = n + 1
        endif
   52   continue
      endif
      backspace(11)
      read (11,'(a80)') ju80
      write (13,'(a80)') ju80
   51 read (11,'(20a1)') (m(i),i=1,20)
      nomat=0
      nomat2=0
      do 53 i=1,5
   53    if (m(i).ne.' ') nomat=1
      do 54 i=1,20
   54    if (m(i).ne.' ') nomat2=1
      if (nomat.eq.1.or.nomat2.eq.0) then
         backspace (11)
         goto 40
      else
         backspace (11)
         read (11,'(a80)') ju80
         write (13,'(a80)') ju80
         goto 51
      endif
50 close(12)
      close (11)
c
c... Create output files and label them.  "mb11t.out" does not depend on
the
```

```
c... material, the others do.  mb13t.out contains only beginning of step
c
      if (frimp.lt.0.) frimp = abs(frimp)
      open (14,file='mb11t.out',status='unknown')
      write (14,'(a72)') title
      write (14,961) pow,days,nouter,ninner,npre,frimp
      write (14,'(a33)') 'Monteburns MCNP k-eff Versus Time'
      write (14,'(a34,a28)') '        days     k-eff     rel err',
     &                   '    nu       avQfis      eta'
      close (14)
c
      open (14,file='mb13t.out',status='unknown')
      write (14,'(/,a42)') 'Monteburns MCNP k-eff at Beginning of Step'
      write (14,'(a34,a6)') '        days     k-eff     rel err',
     &                   '    nu'
      close (14)
c
c... Create file names
c
      do 70 j=1,nmat
      if (j.lt.10) then
        file1t = 'mb1t_'//char(j+48)//'.out'
        file2t = 'mb2t_'//char(j+48)//'.out'
        file3t = 'mb3t_'//char(j+48)//'.out'
        file4a = 'mb4a_'//char(j+48)//'.out'
        file4b = 'mb4b_'//char(j+48)//'.out'
        file5t = 'mb5t_'//char(j+48)//'.out'
        file5x = 'mb5tx_'//char(j+48)//'.out'
        file7t = 'mb7t_'//char(j+48)//'.out'
        file8t = 'mb8t_'//char(j+48)//'.out'
        fil12t = 'mb12t_'//char(j+48)//'.out'
        fil12a = 'mb12a_'//char(j+48)//'.out'
      elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        file1t = 'mb1t_'//char(j1+48)//char(j2+48)//'.out'
        file2t = 'mb2t_'//char(j1+48)//char(j2+48)//'.out'
        file3t = 'mb3t_'//char(j1+48)//char(j2+48)//'.out'
        file4a = 'mb4a_'//char(j1+48)//char(j2+48)//'.out'
        file4b = 'mb4b_'//char(j1+48)//char(j2+48)//'.out'
        file5t = 'mb5t_'//char(j1+48)//char(j2+48)//'.out'
        file5x = 'mb5tx_'//char(j1+48)//char(j2+48)//'.out'
        file7t = 'mb7t_'//char(j1+48)//char(j2+48)//'.out'
        file8t = 'mb8t_'//char(j1+48)//char(j2+48)//'.out'
        fil12t = 'mb12t_'//char(j1+48)//char(j2+48)//'.out'
        fil12a = 'mb12a_'//char(j1+48)//char(j2+48)//'.out'
      endif
c
      open (14,file='mb1',status='unknown')
      write (14,'(/,a29)') 'Monteburns Transport History '
      close (14)
```

```
 open (14,file=file1t,status='unknown')
 write (14,'(/,a29,a12,i3,11x,a14,32x,a20)')
&'Monteburns Transport History ',
&'for material',j,'total material','for actinide    '
 write (14,'(a31,a51,a50,a17)') '      Qfis      Flux      SigmaF',
&         '    Power       Burnup  n,gamma   n,fission fis/cap',
&         '   n2n        eta      n,gamma   n,fission fis/cap',
&         '   n2n        eta'
 close (14)
 open (14,file='mb2',status='unknown')
 write (14,'(/,a41)') 'Monteburns 1-group n,gamma Cross Sections'
 close (14)
 open (14,file=file2t,status='unknown')
 write (14,'(/,a33,a21,i3)')
&'Monteburns 1-group n,gamma Cross ',
&'Sections for material',j
 write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j))
 close (14)
 open (14,file='mb3',status='unknown')
 write (14,'(/,a41)') 'Monteburns 1-group Fission Cross Sections'
 close (14)
 open (14,file=file3t,status='unknown')
 write (14,'(/,a33,a21,i3)')
&'Monteburns 1-group Fission Cross ',
&'Sections for material',j
 write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j))
 close (14)
 open (14,file='mb4a',status='unknown')
 write (14,'(/,a72)') title
 write (14,961) pow,days,nouter,ninner,npre,frimp
 write (14,'(a43)') 'Monteburns Spectrum for Each Predictor Step'
 close (14)
 open (14,file=file4a,status='unknown')
 write (14,'(/,a30,a27,i3)')
&'Monteburns Spectrum for Each ',
&'Predictor Step for material',j
 write (14,'(a63)')
&'        <.1eV      <1eV    <100eV   <100keV     <1MeV    <20MeV'
 close (14)
 open (14,file='mb4b',status='unknown')
 write (14,'(/,a29)') 'Monteburns Grams at Midpoint'
 close (14)
 open (14,file=file4b,status='unknown')
 write (14,'(/,a29,a13,i3)')
&'Monteburns Grams at Midpoint',
&' for material',j
 write (14,'(a40)') '1st row is actual, 2nd row was predicted'
 write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j)),'actinide'
 close (14)
 open (14,file='mb5',status='unknown')
 write (14,'(/,a44)')
```

```
     & 'Monteburns Grams of Material at End of Steps'
      close (14)
      open (14,file='mb12',status='unknown')
      write (14,'(/,a50)')
     & 'Monteburns Grams of Material at Beginning of Steps'
      close (14)
      open (14,file=file5t,status='unknown')
      write (14,'(/,a44,a13,i3)')
     & 'Monteburns Grams of Material at End of Steps',
     & ' for material',j
      write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j)),'actinide'
      close (14)
      open (14,file=fil12t,status='unknown')
      write (14,'(/,a47,a13,i3)')
     & 'Monteburns Grams of Material at Begin. of Steps',
     & ' for material',j
      write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j)),'actinide'
      close (14)
      open (14,file=file5x,status='unknown')
      write (14,'(/,a44,a13,i3)')
     & 'Monteburns Grams of Material at End of Steps',
     & ' for material',j
      write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j)),'actinide'
      close (14)
      open (14,file=fil12a,status='unknown')
      write (14,'(/,a47,a13,i3)')
     & 'Monteburns Grams of Material at Begin.of Steps',
     & ' for material',j
      write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j)),'actinide'
      close (14)
      open (14,file='mb7',status='unknown')
      write (14,'(/,a35,a28,i3)') 'Fractional Importance of Radionuclid'
     & ,'es Sent From ORIGEN2 to MCNP'
      close (14)
      open (14,file=file7t,status='unknown')
      write (14,'(/,a60,/,a20,i3)')
     & 'Fractional Importance of Radionuclides Sent From ORIGEN2 to ',
     & '  MCNP for material',j
      write (14,'(/,a5,a62)') 'step#',
     & '      isotope grams     mass fra  atom fra  capture   fission'
      close (14)
      open (14,file='mb8',status='unknown')
      write (14,'(/,a35,a12,i3)') 'Monteburns Fission-to-Capture Ratio'
      close (14)
      open (14,file=file8t,status='unknown')
      write (14,'(/,a35,a13,i3)')
     & 'Monteburns Fission-to-Capture Ratio',
     & ' for material',j
      write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j))
      close (14)
  961 format (/'Total Power (MW) =',1pe10.2,'   Days =',1pe10.2,/
```

```
     & '# outer steps =',i2,',  # inner steps =',i3,
     & ',  # predictor steps =',i2,/
     & 'Importance Fraction = ',0pf6.4/)
  70 continue
c
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...WTALLY2 writes the tally cards to tal2.inp which is appended
c...to mcnp input file, and creates new mbmc file that does not
c...include tallied materials (run only once at beginning of monteburns)
c
      subroutine wtal2
      character ju6*6,tcell(999,49)*6,ncell*6,file6t*12,file2*12
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      dimension vol(49),ntc(49)
c
c...Determine cells to tally
c
  40 read (5,'(a6)') ju6
     if (ju6.ne.'1cells') goto 40
c
      read (5,'(///)')
  50 read (5,'(i6,a6,i5,1x,1p3e12.5)') n,ncell,nmt,aden,gden,vol1
     do 55 j=1,nmat
     if (nmt.eq.abs(mt(j))) then
       ntc(j)=ntc(j)+1
       tcell(ntc(j),j)=ncell
       vol(j)=vol(j)+vol1
     end if
  55 continue
     if (n.ne.0) goto 50
c
c...Write tally2 file
c
      do 100 j=1,nmat
      if (j.lt.10) then
        file2  = 'tal2_'//char(j+48)//'.inp'
        file6t = 'mb6t_'//char(j+48)//'.out'
      elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        file2  = 'tal2_'//char(j1+48)//char(j2+48)//'.inp'
        file6t = 'mb6t_'//char(j1+48)//char(j2+48)//'.out'
      endif
```

```
      if (voli(j).ne.0.) vol(j)=voli(j)
      if (vol(j).eq.0) then
        write (6,*) '***** MB ERROR: No tally volume'
        stop
      end if
      open (11,file=file2,status='unknown')
c
c...Write energy tallies (tally numbers range from 14 to 494)
c... (1 to 49 represents material number)
c
      write (11,911) (10+j)
  911 format ('c'/'f',i2,'4:n  (')
      do 80 i=1,ntc(j)
   80 write (11,912) tcell(i,j)
  912 format (7x,a6,' ')
      write (11,913)
  913 format (14x,')')
      write (11,915) (10+j),(10+j),vol(j),(10+j)
  915 format ('fc',i2,'4  MonteBurns Energy Spectrum Tallies'/
     &  'sd',i2,'4  ',1pe12.5/
     &  'e',i2,'4    1.0e-7 1.0e-6 1.0e-4 1.0e-1 1.0 20.0')
c
c...Write header for xs tallies
c
      write (11,911) (50+j)
      do 90 i=1,ntc(j)
   90 write (11,922) tcell(i,j)
  922 format (7x,a6,' ')
      write (11,923)
  923 format (14x,')')
      write (11,924) (50+j),(50+j),vol(j),(50+j)
  924 format ('fc',i2,'4  MonteBurns Cross Section Tallies'/
     &  'sd',i2,'4  ',1pe12.5/'fm',i2,'4   (1)')
c
      open (14,file='mb6',status='unknown')
      write (14,'(/,a24)') 'Monteburns Flux Spectrum'
      close (14)
      open (14,file=file6t,status='unknown')
      write (14,'(/,a25,a12,i3)') 'Monteburns Flux Spectrum ',
     & 'for material',j
      write (14,'(a63)')
     & '        <.1eV      <1eV    <100eV   <100keV     <1MeV    <20MeV'
      close (14)
  100 continue
      close (11)
c
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
```

```
c...WTALLY writes the tally cards to tal1.inp and tal3.inp which
c...are appended to mcnp input file
c
      subroutine wtally
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      character file1*12,file3*12
c
c...Write tally files 1 and 3
c
      ii = 100
      ij = 900
      do 100 j=1,nmat
      iflag = 0
      if (j.lt.10) then
        file1 = 'tal1_'//char(j+48)//'.inp'
        file3 = 'tal3_'//char(j+48)//'.inp'
      elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        file1 = 'tal1_'//char(j1+48)//char(j2+48)//'.inp'
        file3 = 'tal3_'//char(j1+48)//char(j2+48)//'.inp'
      endif
      open (11,file=file1,status='unknown')
      open (12,file=file3,status='unknown')
      do 90 i=1,ntot(j)
      ii=ii+1
      write (11,901) ii,niso(i,j)
  901 format ('m',i3,4x,a10,'   1.0')
c
c   Equate (n,t) reaction to (n,alpha) for Lithium-6
c   All others are true (n,alpha) cross sections
c
      if (nisn(i,j).eq.3006) then
        write (12,920) ii
      elseif (nisn(i,j).lt.89000) then
        write (12,921) ii
      elseif (nisn(i,j).ge.89000) then
        iflag = 1
        write (12,922) ii
      endif
   90 continue
      ij = ij + 1
      if (iflag.eq.1) write (12,923) ij
      write (12,923) abs(mt(j))
      close (11)
  100 continue
c
```

```
c       -2 is the total capture cross section
c       16 is (n,2n) cross section
c       105 is (n,t) cross section
c       107 is (n,alpha) cross section
c       103 is (n,p) cross section (for activation products)
c       17 is (n,3n) cross section
c       -6 is the total fission cross section (for actinides)
c       452 is nu bar - only used for verification purposes
c
  920 format (8x,'(1 ',i3,' (102)  (16)  (105) (103))')
  921 format (8x,'(1 ',i3,' (102)  (16)  (107) (103))')
  922 format (8x,'(1 ',i3,' (102)  (16)  (17)  (-6))')
  923 format (8x,'(1 ',i3,' (-2)  (16)  (452) (-6))')
c
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...WORCOMP writes composition input file for origen fort.7, which is
c...read by origen as fort.4.  Units are g-atoms (grams / atomic mass)
c...(one time execute at beginning of monteburns)
c
      subroutine worcom
c
      dimension nuc(99,49),f(99,49),gden(49),vol(49),nc(99),fn(99)
      dimension ij(49),nelem(999,49),nisop(999,49),atomfr(999,49,20),
     & nisot(999,49,20),naix(999,49,20),iflag(999,49),gmat(999,49,20),
     & aix(999,49,20)
      character ju6*6,ju10*10,met*1,ninat*10,fname*12,fnat*12,
     &          fmcnp*12,nmcnp*20
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
c
   10 read (5,'(32x,a10)') ju10
      if (ju10.ne.'mass fract') goto 10
c
c...Read mass fractions for material
c
      ifd = 0
      read (5,*)
   20 read (5,'(i6,5x,4(6x,i5,2x,1pe11.5))')
     &  mtn,(nc(i),fn(i),i=1,4)
      ii = 0
      im = 0
      do 25 j=1,nmat
      if (mtn.eq.abs(mt(j))) then
        do 22 i=1,4
```

```
            nuc(i,j) = nc(i)
      22    f(i,j) = fn(i)
            nmt = j
            im = 1
            ifd = ifd + 1
          endif
      25  continue
          if (im.eq.0) goto 20
c
      30  ii=ii+4
          read (5,'(a6)') ju6
          if (ju6.eq.'1cells') then
             ii=ii-4
             ij(nmt) = ii
             goto 42
          else
             backspace (5)
          endif
          read (5,'(i6,5x,4(6x,i5,2x,1pe11.5))')
         &  mtn,(nuc(i,nmt),f(i,nmt),i=1+ii,4+ii)
          if (mtn.gt.0.and.mtn.ne.abs(mt(nmt))) ii=ii-4
          if (mtn.eq.0.and.nuc(4+ii,nmt).ne.0) goto 30
          ij(nmt) = ii
          if (ifd.ne.nmat) then
           backspace(5)
           goto 20
          endif
c
c...Determine gram density and volume of cells (for now just 1)
c
      40  read (5,'(a6)') ju6
          if (ju6.ne.'1cells') goto 40
c
      42  read (5,'(///)')
      50  read (5,'(2i6,i5,1x,1p3e12.5)') n,ncell,nmt,aden,gden1,vol1
          do 51 j=1,nmat
          if (nmt.eq.abs(mt(j))) then
            vol(j)=vol(j)+vol1
            gden(j)=gden1
          end if
      51  continue
          if (n.ne.0) goto 50
c
c...Make sure isos have been read correctly, erase spurios readings
c
          do 80 j=1,nmat
          do 52 i=1,4+ij(j)
          nogo=0
          if (nuc(i,j).lt.1000) nogo=1
          if (nogo.eq.1) nuc(i,j)=0
      52  continue
```

```
c
c...Write grams of material to fort.7 (origen comp file) or mnat.tmp
c...if a natural iso appears in mcnp input file
c
      if (voli(j).ne.0.) vol(j)=voli(j)
      voli(j) = vol(j)
      call wmbinp
c
      if (j.lt.10) then
        fnat = 'mnat_'//char(j+48)//'.tmp'
        fname = 'fort_'//char(j+48)//'.7'
      elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        fnat = 'mnat_'//char(j1+48)//char(j2+48)//'.tmp'
        fname = 'fort_'//char(j1+48)//char(j2+48)//'.7'
      endif
      open (11,file=fname,status='unknown')
      open (12,file=fnat,status='unknown')
      do 58 i=1,4+ij(j)
      iflag(i,j) = 0
      if (nuc(i,j)-1000*(nuc(i,j)/1000).eq.0.and.nuc(i,j).gt.0) then
        open (16,file='natelem',status='unknown')
        read (16,*)
        read (16,*)
   54   read (16,*) nelem(i,j)
        read (16,*) nisop(i,j)
        do 56 n=1,nisop(i,j)
   56   read (16,'(i5,3x,f10.5)',err=56,end=53)
     &        nisot(i,j,n),atomfr(i,j,n)
        if (nelem(i,j).eq.nuc(i,j)/1000) then
          iflag(i,j) = 1
          goto 53
        else
          goto 54
        endif
   53   close (16)
        open (13,file='mbxs.inp',status='unknown')
        ifd=0
   55   read (13,*,end=57) nixs
        if (nixs.eq.nuc(i,j)) ifd=1
        if (ifd.eq.0) goto 55
        backspace (13)
        read (13,'(a10)') ninat
        write (12,'(i2,4x,a10)') nelem(i,j),ninat
   57   if (ifd.eq.0) write (6,*)
     &  '***** MB WARNING: Natural iso xs not found ',nuc(i,j)
        close (13)
      elseif (nuc(i,j).ne.0) then
       if (j.lt.10) then
        fmcnp = 'mcnp_'//char(j+48)//'.inp'
```

```
        elseif (j.ge.10) then
         j1 = j/10
         j2 = j - j1*10
         fmcnp = 'mcnp_'//char(j1+48)//char(j2+48)//'.inp'
        endif
        open (17,file=fmcnp,status='unknown')
        open (13,file='mbxs.inp',status='unknown')
        ifd=0
   66   read (13,*,end=67) nixs
        if (nixs.eq.nuc(i,j)) ifd=1
        if (ifd.eq.0) goto 66
        backspace (13)
        read (13,'(a10)') nmcnp
        write (17,'(a5,2x,a10)') nmcnp(1:5),nmcnp
   67   if (ifd.eq.0) write (6,*)
     &   '***** MB WARNING: Iso xs not found ',nuc(i,j)
        close (13)
       end if
   58 continue
       close (12)
       close (17)
c
c...Write non-actinides to fort.7, sort numerically for xs file read
c
       do 65 k=1,4+ij(j)
       nmin=99999
       ni=0
       do 60 i=1,4+ij(j)
       a=float(nuc(i,j))-float(1000*(nuc(i,j)/1000))
       if (nuc(i,j).lt.83000.and.nuc(i,j).gt.1000) then
         if (nuc(i,j).lt.nmin) then
           nmin=nuc(i,j)
           if (iflag(i,j).ne.1) then
              ai=a
           else
              do 59 n=1,nisop(i,j)
              naix(i,j,n)=nisot(i,j,n) - 1000*(nisot(i,j,n)/1000)
   59         aix(i,j,n) = float(nisot(i,j,n))
     &             - float(1000*(nisot(i,j,n)/1000))
           endif
           ni=i
         end if
       end if
   60 continue
       if (ni.gt.0) then
         kxs=1
         met='0'
         if (iflag(ni,j).eq.1) then
           do 62 n=1,nisop(ni,j)
           gmat(ni,j,n) = f(ni,j)*gden(j)*vol(j)/aix(ni,j,n)
           gmat(ni,j,n) = gmat(ni,j,n)*atomfr(ni,j,n)
```

```
            if (naix(ni,j,n).lt.10) then
              write (11,912) kxs,nelem(ni,j),naix(ni,j,n),met,gmat(ni,j,n)
            elseif (naix(ni,j,n).lt.100) then
              write (11,913) kxs,nelem(ni,j),naix(ni,j,n),met,gmat(ni,j,n)
            else
              write (11,914) kxs,nelem(ni,j),naix(ni,j,n),met,gmat(ni,j,n)
            endif
   62       continue
          else
            gma=f(ni,j)*gden(j)*vol(j)/ai
            write (11,911) kxs,nuc(ni,j),met,gma
          endif
          nuc(ni,j)=0
         end if
   65 continue
c
c...Write actinides to fort.7, sort numerically for xs file read
c
      do 75 k=1,4+ij(j)
      nmin=99999
      ni=0
      do 70 i=1,4+ij(j)
      a=float(nuc(i,j))-float(1000*(nuc(i,j)/1000))
      if (nuc(i,j).ge.83000.and.a.gt.0.) then
        if (nuc(i,j).lt.nmin) then
          nmin=nuc(i,j)
          if (iflag(i,j).ne.1) then
             ai=a
          else
             do 69 n=1,nisop(i,j)
             naix(i,j,n)=nisot(i,j,n) - 1000*(nisot(i,j,n)/1000)
   69        aix(i,j,n) = float(nisot(i,j,n))
     &            - float(1000*(nisot(i,j,n)/1000))
          endif
          ni=i
        end if
      end if
   70 continue
      if (ni.gt.0) then
        kxs=2
        met='0'
        if (nuc(ni,j).eq.95242) met='1'
        if (iflag(ni,j).eq.1) then
          do 72 n=1,nisop(ni,j)
          gmat(ni,j,n) = f(ni,j)*gden(j)*vol(j)/aix(ni,j,n)
          gmat(ni,j,n) = gmat(ni,j,n)*atomfr(ni,j,n)
          if (naix(ni,j,n).lt.10) then
            write (11,912) kxs,nelem(ni,j),naix(ni,j,n),met,gmat(ni,j,n)
          elseif (naix(ni,j,n).lt.100) then
            write (11,913) kxs,nelem(ni,j),naix(ni,j,n),met,gmat(ni,j,n)
          else
```

```
            write (11,914) kxs,nelem(ni,j),naix(ni,j,n),met,gmat(ni,j,n)
          endif
   72     continue
        else
          gma=f(ni,j)*gden(j)*vol(j)/ai
          write (11,911) kxs,nuc(ni,j),met,gma
        endif
        nuc(ni,j)=0
       end if
   75 continue
      write (11,'(a12)') '0  0  0  0'
      close (11)
  911 format (i4,i6,a1,1pe12.4,
     & '      0  0.0000E+00      0  0.0000E+00      0  0.0000E+00')
  912 format (i4,i3,'00',i1,a1,1pe12.4,
     & '      0  0.0000E+00      0  0.0000E+00      0  0.0000E+00')
  913 format (i4,i3,'0',i2,a1,1pe12.4,
     & '      0  0.0000E+00      0  0.0000E+00      0  0.0000E+00')
  914 format (i4,i3,i3,a1,1pe12.4,
     & '      0  0.0000E+00      0  0.0000E+00      0  0.0000E+00')
   80 continue
c
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...WORINP writes the origen input files.
c...put GTO 9 card 1/2 way for predictor step.
c...Do not write over restart files
c
      subroutine worinp
c
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      character libnam*80,xslib*6,lib(99)*10,decay1*80
      character ju3*3,fname*22,flname*22,file9t*12,dec80*80
      integer end,olibn
      dimension day(99),nfeed(99,49),gf1(99,49),gf2(99,49),mfeed(10),
     & kfeed(10),kfeed1(10,30),kfeed2(10,30),tmst(99),
     & ifeed(10,30),ffeed(10,30),tfeed(999,49),ttfeed(999,49),
     & nf1(99,49),rf(99,49),pfra(99),lb(99,4),nmt(49)
c
      itwo=2
      if (olib(2:2).eq.' ') olibn=ichar(olib(1:1))-48
      if (olib(2:2).ne.' ') olibn=(ichar(olib(1:1))-48)*10+
     & ichar(olib(2:2))-48
c
```

```
c...Add cross section values to existing fort.9 file, which previously
only
c...contained the origen2 decay library.
c
      open (15,file='fort.9.0',status='unknown')
      xslib(1:4)='orig'
      xslib(5:6) = olib(1:2)
      do 2 i=72,1,-1
        if (locale(i:i).eq.' ') end=i-1
    2 continue
      decay1 = locale(1:end)//'/orig21'
      open (12,file=decay1,status='old')
    3 read (12,'(a80)',end=4) dec80
      write (15,'(a80)') dec80
      goto 3
    4 libnam=locale(1:end)//'/'//xslib
      open (18,file=libnam,status='old')
    5 read (18,'(a80)',end=6) dec80
      write (15,'(a80)') dec80
      goto 5
    6 close(12)
      close(15)
      close(16)
c
c...Create data file from scratch ; First read feed rate data file
c
      if(days.eq.0.0) then
      nfd = 1
      open (11,file='feed',status='old')
c
c...First read the two lines of headings
c
      read (11,*)
      read (11,*)
      do 8 i=1,nouter
        do 7 j=1,nmat
        if (j.eq.1) then
         read (11,*) tmst(i),day(i),
     &   pfra(i),nmt(1),nfeed(i,1),gf1(i,1),gf2(i,1),nf1(i,1),rf(i,1)
        elseif (j.ge.2) then
         read (11,*) nmt(j),
     &   nfeed(i,j),gf1(i,j),gf2(i,j),nf1(i,j),rf(i,j)
        endif
       ndisc = 0
       if (gf1(i,j).eq.-2.) ndisc = 1
       if (j.lt.10) then
        if (i.lt.10) then
         flname = './tmpfile/param_'//char(j+48)//'.'//char(i+48)
        elseif (i.ge.10) then
         i1 = i/10
         i2 = i - i1*10
```

```
         flname = './tmpfile/param_'//char(j+48)//'.'
     &                //char(i1+48)//char(i2+48)
        endif
       elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        if (i.lt.10) then
         flname = './tmpfile/param_'//char(j1+48)//char(j2+48)//'.'
     &               //char(i+48)
        elseif (i.ge.10) then
         i1 = i/10
         i2 = i - i1*10
         flname = './tmpfile/param_'//char(j1+48)//char(j2+48)//'.'
     &               //char(i1+48)//char(i2+48)
        endif
       endif
       open (16,file=flname,status='unknown')
       write (16,910) ndisc
  910  format (i4,' ndisc')
       close (16)
    7  if (i.gt.1.and.gf1(i,j).eq.-1.) gf1(i,j)=gf2(i-1,j)
    8  days=days+day(i)
       read (11,'(i4)') nfs
       do 9 n=1,nfs
       read (11,'(i4)') mfeed(n)
       do 9 m=1,mfeed(n)
    9  read (11,'(i5,f9.7)') ifeed(n,m),ffeed(n,m)
       read (11,'(i4)') nrs
       do 10 n=1,nrs
       read (11,'(i4)') kfeed(n)
       do 10 k=1,kfeed(n)
   10  read (11,'(i4,i4)') kfeed1(n,k),kfeed2(n,k)
c
c...Rewrite mb.inp with new days (later add feed data to output)
c
       call wmbinp
       else
         do 42 i=1,nouter
   42    day(i) = days/float(nouter)
       endif
   45  continue
       close (11)
c
c...Write flag to file that indicates whether a feed file
c...exists or not
c
       open (17,file='./tmpfile/params2',status='unknown')
       write (17,950) nfd
  950  format (i4,' nfd')
       close (17)
c
```

```
c...Write origen input file for each step and write feed data to mb9.out
c
      do 100 j=1,nmat
      if (j.lt.10) then
        file9t = 'mb9t_'//char(j+48)//'.out'
      elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        file9t = 'mb9t_'//char(j1+48)//char(j2+48)//'.out'
      endif
      open (14,file='mb9',status='unknown')
      write (14,'(/,a21)') 'Monteburns Inventory '
      close (14)
      open (14,file=file9t,status='unknown')
      write (14,'(/,a33,a13,i3)')
     & 'Monteburns Grams of Feed per Step',
     & ' for material',j
      write (14,'(a5,2x,a4,5x,a9,30(1x,a9))')
     &            'mat #','days',(niso(i,j),i=1,nauto(j)),'actinide'
      do 48 i=1,nouter
      zero= 0.0
c
c...If restart read flux from old mbori and put in new mbori
c
      if (i.eq.nrst+1.and.nrst.gt.0) then
       if (j.lt.10) then
        if (i.lt.10) then
          fname='./tmpfile/mbori_'//char(j+48)//'.'//char(i+48)
        elseif (i.ge.10) then
          i1 = i/10
          i2 = i - i1*10
          fname='./tmpfile/mbori_'//char(j+48)//'.'
     &           //char(i1+48)//char(i2+48)
        end if
       elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        if (i.lt.10) then
          fname='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &           //char(i+48)
        elseif (i.ge.10) then
          i1 = i/10
          i2 = i - i1*10
          fname='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &           //char(i1+48)//char(i2+48)
        endif
       endif
       open (11,file=fname,status='unknown')
   12  read (11,'(a3)',end=14) ju3
       if (ju3.ne.'IRF') goto 12
       backspace (11)
```

```
        read (11,900) zero
   14   close (11)
  900   format (19x,1pe13.5)
        end if
c
   13 n=nfeed(i,j)
      dstep=day(i)/float(ninner)
      do 15 m=1,nauto(j)+1
   15 tfeed(m,j)=0.
c
      if (j.lt.10) then
       if (i.lt.10) then
        fname='./tmpfile/mbori_'//char(j+48)//'.'//char(i+48)
       elseif (i.ge.10) then
        i1 = i/10
        i2 = i - i1*10
        fname='./tmpfile/mbori_'//char(j+48)//'.'
     &         //char(i1+48)//char(i2+48)
       end if
      elseif (j.ge.10) then
       j1 = j/10
       j2 = j - j1*10
       if (i.lt.10) then
        fname='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &         //char(i+48)
       elseif (i.ge.10) then
        i1 = i/10
        i2 = i - i1*10
        fname='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &         //char(i1+48)//char(i2+48)
       endif
      endif
      open (13,file=fname,status='unknown')
c
c...Write group info to new file, then write initial commands.
c
      data (lb(22,ii),ii=1,3),lib(22) /204,205,206,'PWRU'/
      data (lb(23,ii),ii=1,3),lib(23) /207,208,209,'PWRPUU'/
      data (lb(24,ii),ii=1,3),lib(24) /210,211,212,'PWRPUPU'/
      data (lb(25,ii),ii=1,3),lib(25) /213,214,215,'PWRDU3TH'/
      data (lb(26,ii),ii=1,3),lib(26) /225,226,227,'PWRD5D33'/
      data (lb(27,ii),ii=1,3),lib(27) /222,223,224,'PWRD5D35'/
      data (lb(28,ii),ii=1,3),lib(28) /216,217,218,'PWRPUTH'/
      data (lb(29,ii),ii=1,3),lib(29) /219,220,221,'PWRU50'/
      data (lb(30,ii),ii=1,3),lib(30) /251,252,253,'BWRU'/
      data (lb(31,ii),ii=1,3),lib(31) /254,255,256,'BWRPUU'/
      data (lb(32,ii),ii=1,3),lib(32) /257,258,259,'BWRPUPU'/
      data (lb(33,ii),ii=1,3),lib(33) /201,202,203,'THERMAL'/
      data (lb(34,ii),ii=1,3),lib(34) /401,402,403,'CANDUNAU'/
      data (lb(35,ii),ii=1,3),lib(35) /404,405,406,'CANDUSEU'/
      data (lb(36,ii),ii=1,3),lib(36) /311,312,313,'AMOPUUUC'/
```

```
      data (lb(37,ii),ii=1,3),lib(37) /314,315,316,'AMOPUUUA'/
      data (lb(38,ii),ii=1,3),lib(38) /317,318,319,'AMOPUUUR'/
      data (lb(39,ii),ii=1,3),lib(39) /301,302,303,'EMOPUUUC'/
      data (lb(40,ii),ii=1,3),lib(40) /304,305,306,'EMOPUUUA'/
      data (lb(41,ii),ii=1,3),lib(41) /307,308,309,'EMOPUUUR'/
      data (lb(42,ii),ii=1,3),lib(42) /321,322,323,'AMORUUUC'/
      data (lb(43,ii),ii=1,3),lib(43) /324,325,326,'AMORUUUA'/
      data (lb(44,ii),ii=1,3),lib(44) /327,328,329,'AMORUUUR'/
      data (lb(45,ii),ii=1,3),lib(45) /331,332,333,'AMOPUUTC'/
      data (lb(46,ii),ii=1,3),lib(46) /334,335,336,'AMOPUUTA'/
      data (lb(47,ii),ii=1,3),lib(47) /337,338,339,'AMOPUUTR'/
      data (lb(48,ii),ii=1,3),lib(48) /341,342,343,'AMOPTTTC'/
      data (lb(49,ii),ii=1,3),lib(49) /344,345,346,'AMOPTTTA'/
      data (lb(50,ii),ii=1,3),lib(50) /347,348,349,'AMOPTTTR'/
      data (lb(51,ii),ii=1,3),lib(51) /361,362,363,'AMOITTTC'/
      data (lb(52,ii),ii=1,3),lib(52) /364,365,366,'AMOITTTA'/
      data (lb(53,ii),ii=1,3),lib(53) /367,368,369,'AMOITTTR'/
      data (lb(54,ii),ii=1,3),lib(54) /371,372,373,'AMO2TTTC'/
      data (lb(55,ii),ii=1,3),lib(55) /374,375,376,'AMO2TTTA'/
      data (lb(56,ii),ii=1,3),lib(56) /377,378,379,'AMO2TTTR'/
      data (lb(57,ii),ii=1,3),lib(57) /351,352,353,'AMOXTTTC'/
      data (lb(58,ii),ii=1,3),lib(58) /354,355,356,'AMOXTTTA'/
      data (lb(59,ii),ii=1,3),lib(59) /357,358,359,'AMOXTTTR'/
      data (lb(60,ii),ii=1,3),lib(60) /381,382,383,'FFTFC'/
      data (lb(65,ii),ii=1,3),lib(65) /381,382,383,'ADV3'/
      data (lb(66,ii),ii=1,3),lib(66) /204,205,206,'PWRSPEC'/
c
      write (13,921)
      nn = abs(nf1(i,j))
      if(nf1(i,j).le.0) then
          write (13,921)
          goto 19
      endif
      do 16 m=1,9
   16 write (13,918) m
      do 17 m=10,14
   17 write (13,922) m
      write (13,920) (1.0 - rf(i,j))
      write (13,921)
      if(nn.gt.nrs) then
        write (6,919) i
  919   format ('***** MB:  Invalid removal group ',
     &               'entered for outer step number',i4)
        stop
      endif
      do 18 k=1,kfeed(nn)
      do 18 m=abs(kfeed1(nn,k)),abs(kfeed2(nn,k))
   18 write (13,923) m
   19 write (13,921)
      write (13,924)
      write (13,925)
```

```
      write (13,926) lb(olibn,1),lb(olibn,2),lb(olibn,3),lib(olibn)
      write (13,927) lb(olibn,1),lb(olibn,2),lb(olibn,3)
      write (13,928)
      write (13,929)
      write (13,930)
      write (13,931)
      write (13,932)
      write (13,933)
      write (13,934)
      write (13,935)
      write (13,905)
      kk=2
      icont=0
      if (n.gt.0.and.gf1(i,j).ne.-2) icont=1
      if (icont.eq.1) kk=10
c
c... Write various loops into origen input file
c
      do 22 k=1,kk
        write (13,901)
        if (icont.eq.1) then
         write (13,911)
         write (13,902) ninner/10
         dburn=dstep*float(ninner/10)
        else
         write (13,902) ninner/2
         dburn=dstep*float(ninner/2)
        end if
        write (13,903) dstep,zero
        write (13,904)
        write (13,905)
c
        if (n.gt.0) then
        do 21 m=1,mfeed(n)
         nm=2
         if (ifeed(n,m).lt.89000) nm=1
         ifd6=ifeed(n,m)*10
         if (ifd6.eq.952420) ifd6=ifd6+1
         if (gf1(i,j).ne.-2) then
            gfs=(float(k)-.5)/float(kk)*(gf2(i,j)-gf1(i,j))+gf1(i,j)
            gfeed=ffeed(n,m)*gfs*dburn
         else
            if (k.eq.1) then
              gfeed=ffeed(n,m)*gf2(i,j)*day(i)
            else
              gfeed = 0.0
            endif
         endif
         if (ifeed(n,m).ge.89000) then
           tfeed(nauto(j)+1,j)=tfeed(nauto(j)+1,j)+gfeed
         endif
```

```
          do 29 mm=1,nauto(j)
          if (ifd6.eq.nisnr(mm,j))  then
             tfeed(mm,j)=tfeed(mm,j)+gfeed
          endif
   29     continue
c
          if (icont.eq.1) write (13,913) nm,ifd6,ffeed(n,m)*gfs
   21     continue
          if (icont.eq.1) write (13,914)
         end if
c
c... Write end of run 1/2 way through for predictor step
c
         ihalf = 0
         if (k.eq.5) ihalf = 1
         if (k.eq.1.and.icont.eq.0) ihalf=1
         if (ihalf.eq.1) then
          write (13,938)
          if (nf1(i,j).gt.0) write (13,936)
          write (13,937)
          write (13,939)
         end if
c
   22  continue
c
c     complete end of origen input file
c
      if(nf1(i,j).gt.0) write (13,936)
      write (13,937)
c
   25 close (13)
      write (14,'(i2,1x,f8.2,3x,1pe9.2,30e10.2)')
     &                 i,day(i),(tfeed(m,j),m=1,nauto(j)+1)
      do 46 m=1,nauto(j)+1
   46 ttfeed(m,j)=ttfeed(m,j)+tfeed(m,j)
   48 continue
      write (14,'(a3,f8.2,3x,1pe9.2,30e10.2)')
     &           'tot',days,(ttfeed(m,j),m=1,nauto(j)+1)
      write (14,'(/,a41,a13,i3)')
     & 'Monteburns Grams Produced (or Destroyed) per Step',
     & ' for material',j
      write (14,'(3x,a9,30(1x,a9))')
     &           (niso(i,j),i=1,nauto(j)),'actinide'
      close (14)
c
  901 format ('BUP')
  902 format ('DOL   1  ',i4)
  903 format ('IRF   ',1p2e13.5,'    2  3  4  1')
  904 format ('MOV   3 2 0 1.0'/'CON   1'/'BUP')
  905 format ('STP 2')
  911 format ('INP   1  0  1 -1  4  4')
```

```
  913 format (i1,i8,1pe12.4,'  0  0.0')
  914 format ('0')
  918 format (i1,t4,'1 1.0')
  920 format ('15',t4,'1 ',f7.3)
  921 format ('-1')
  922 format (i2,t4,'1 1.0')
  923 format (i2,'  15')
  924 format ('TIT     ORIGEN2 input file for monteburns')
  925 format ('LIP     0 1 0')
  926 format ('RDA *** Libs ',i3,',',i3,',',i3,' = ',a10)
  927 format ('LIB     0 1 2 3 ',3(i3,1x),'9 3 0 3 0')
  928 format ('RDA     1 Bundle of fuel',/,
     &        'RDA   Read initial comps into vector 1 from fort.4 in ',
     &                'gram-atoms')
  929 format ('INP     1 -2 0 -1 4 4')
  930 format ('MOV     1 2 0 1.0',/,
     &        'MOV     1 3 0 0.0',/,
     &        'MOV     1 4 0 0.0')
  931 format ('RDA ***',/,
     &        'RDA *** Set output options (print in grams)')
  932 format ('HED     1  INITIAL')
  933 format ('CUT     5  1.0-10  -1')
  934 format ('OPTA    4*8 7 19*8',/,
     &        'OPTL    4*8 7 19*8',/,
     &        'OPTF    4*8 7 19*8')
  935 format ('RDA ***',/,
     &        'RDA    Begin burn, add cards after STP 2, remove FP at',
     &               ' end of burn')
  936 format ('PRO     2  3  4  -1')
  937 format ('MOV     3  2  0  1.0',/,
     &        'OUT     4  1  1  0  ',/,
     &        'PCH     2  2  2     ',/,
     &        'RDA                 ',/,
     &        'END                 ')
  938 format ('RDA First of 1/2 way predictor cards')
  939 format ('RDA Last of 1/2 way predictor cards')
c
  100 continue
      return
      end
c
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...WORXS calculates new xs from mcnp and modifies the cross
c...sections in fort.9.  Also calculates flux and modifies mbori
c...for 1/2 step
c
      subroutine worxs
c
      character ju10*10,ju80*80,ju3*3,fort7*12,ju6*6,blanks*4,mtuf*20
```

```
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      character tal*3,nm*2,file8*12,file6*12,fname*21,fname2*25
      character fort9*12,f9tmp*15,file1*12,file2*12,file3*12,file1t*12
      dimension xs(999,4,49),eflx(7,49),day(99),nfeed(99,49),flx(49),
     &          gf1(99,49),gf2(99,49),nf1(99,49),rf(99,49),pfra(99),
     &          nmt(49),nisq(49),gad(49),fismac(49),tmst(99),
     &          qfis(49),flux(49),flux2(49),fiscap(999),vol(49),pwr(49),
     &          aval(4,49),absmac(49),frfast(49),frth(49),n2nmac(49),
     &          burnup(0:99,49),fluxy(49),fluxy2(49)
      real keff,nu,macfis,macabs,macn2n,kinfin(49),kinf,mtu(49)
c
c...First obtain data from feed input file
c
      open (17,file='./tmpfile/params2',status='old')
      read (17,'(i4)') nfd
      if (nfd.eq.1) then
        open (11,file='feed',status='old')
        read (11,*)
        read (11,*)
        daynum = 0.0
        do 10 i=1,nouter
         do 5 j=1,nmat
          if (j.eq.1) then
           read (11,*) tmst(i),day(i),pfra(i),
     &      nmt(1),nfeed(i,1),gf1(i,1),gf2(i,1),nf1(i,1),rf(i,1)
          elseif (j.ge.2) then
           read (11,*)
     &      nmt(j),nfeed(i,j),gf1(i,j),gf2(i,j),nf1(i,j),rf(i,j)
          endif
    5    continue
         if (i.eq.nrst+1) goto 15
   10    daynum = daynum + day(i)
      else
         do 12 i=1,nouter
           day(i) = days/float(nouter)
           pfra(i) = 1.0
           if (i.eq.nrst+1) goto 15
   12      daynum = daynum + day(i)
      endif
c
c...Read mass fraction section to get volume of each material
c
      do 13 j=1,nmat
   13 vol(j) = 0.0
   15 close (11)
      open (11,file='mbmco',status='old')
  280 read (11,'(a6)') ju6
```

```
      if (ju6.ne.'1cells') goto 280

      read (11,'(///)')
  290 read (11,'(i6,a6,i5,1x,1p3e12.5)') n,ncell,nt,aden,gden,vol1
      do 295 j=1,nmat
      if (nt.eq.abs(mt(j))) then
        vol(j)=vol(j)+vol1
      end if
  295 continue
      if (n.ne.0) goto 290
c
c...Read keff and calculate nu
c
   20 read (11,'(a10)') ju10
      if (ju10.ne.' neutron c') goto 20
c
      read (11,'(/)')
      read (11,'(31x,1pe10.4)') src
      read (11,'(/////////////)')
      read (11,'(31x,1pe10.4,54x,1pe10.4)') fsrc,floss
      read (11,'(/////)')
      read (11,'(35x,1pe10.4,0p,f7.4)') fmult,err
c
      if (fsrc.ne.0.) then
         nu=fsrc/floss
         keff = (fmult-1.)/(fmult-1./nu)
         relerr = (fmult*(1.+err)-1.)/(fmult*(1.+err)-1./nu)
         relerr = (relerr - keff)/keff
      else
   30    read (11,'(a10)') ju10
         if (ju10.ne.' ----------') goto 30
         read (11,'(/72x,f7.5,41x,f7.5)') keff,relerr
         nu=keff*src/floss
      endif
c
c...Read energy spectrum tallies (if tallies don't exist in mbmco,
c...then tal='yes' (used in later commands)
c
      if (posit.eq.'m') then
      do 68 j=1,nmat
   55 read (11,'(a10)',end=67) ju10
      if (ju10(1:6).ne.'1tally') goto 55
      tal = 'yes'
      mat = 0
      do 60 m=1,nmat
      if (m.ge.10) then
        m1 = m/10
        m2 = m - m1*10
        m1 = m1 + 1
        nm = char(m1+48)//char(m2+48)
      elseif (m.lt.10) then
```

```
        nm = '1'//char(m+48)
      endif
      if (nm.eq.ju10(8:9)) then
       mat = m
       goto 61
      endif
   60 continue
      if (mat.eq.0) goto 55
c
   61 read (11,'(a10)') ju10
      if (ju10.ne.'      ener') goto 61
c
      do 65 i=1,7
   65 read (11,'(17x,1pe11.5)') eflx(i,mat)
      frfast(mat) = 0.
      frth(mat) = (eflx(1,mat)+eflx(2,mat))/eflx(7,mat)
      do 66 i=3,6
   66 frfast(mat) = frfast(mat) + eflx(i,mat)/eflx(7,mat)
c
      if (mat.lt.10) then
         file6 = 'mb6_'//char(mat+48)//'.out'
      elseif (mat.ge.10) then
         j1 = mat/10
         j2 = mat - j1*10
         file6 = 'mb6_'//char(j1+48)//char(j2+48)//'.out'
      endif
      open (14,file=file6,status='unknown')
      write (14,'(i2,1x,6f10.2)') nrst,
     &                  (100.*eflx(i,mat)/eflx(7,mat),i=1,6)
      close (14)
      goto 68
   67 write (6,*) '***** MB ERROR:  Not all user-specified MCNP',
     &            ' materials were found in MCNP output file'
      stop
   68 continue
c
c...Read tallies and calculate new cross sections
c
      do 88 j=1,nmat
      iflag = 0
   70 read (11,'(a10)') ju10
      if (ju10(1:6).ne.'1tally') goto 70
      mat = 0
      do 72 m=1,nmat
      if (m.ge.10) then
        m1 = m/10
        m2 = m - m1*10
        m1 = m1 + 5
        nm = char(m1+48)//char(m2+48)
      elseif (m.lt.10) then
        nm = '5'//char(m+48)
```

```
      endif
      if (nm.eq.ju10(8:9)) then
        mat = m
        goto 74
      endif
   72 continue
      if (mat.eq.0) goto 70
c
   74 read (11,'(a10)') ju10
      if (ju10.ne.' multiplie') goto 74
c
      read (11,'(17x,1pe11.5)') flx(mat)
      if (flx(mat).eq.0) write (6,*) '***** MB: Tally read error'
      do 80 i=1,ntot(j)
      do 80 m=1,4
   76 read (11,'(a10)') ju10
      if (ju10.ne.' multiplie') goto 76
      read (11,'(17x,1pe11.5)') xs(i,m,j)
      xs(i,m,j)=xs(i,m,j)/flx(j)
      if (nisn(i,j).ge.89000) iflag = 1
   80 continue
      if  (iflag.eq.1) then
       do 85 m=1,4
   82   read (11,'(a10)') ju10
        if (ju10.ne.' multiplie') goto 82
        read (11,'(17x,1pe11.5)') xs(ntot(j)+1,m,j)
        xs(ntot(j)+1,m,j)=xs(ntot(j)+1,m,j)/flx(j)
   85  continue
      endif
      do 87 m=1,4
   86  read (11,'(a10)') ju10
       if (ju10.ne.' multiplie') goto 86
       read (11,'(17x,1pe11.5)') xs(ntot(j)+2,m,j)
       xs(ntot(j)+2,m,j)=xs(ntot(j)+2,m,j)/flx(j)
   87 continue
      if (xs(ntot(j)+2,1,j) + xs(ntot(j)+2,4,j).ne.0.0) then
        kinfin(j) = (nu*xs(ntot(j)+2,4,j) + 2.0*xs(ntot(j)+2,2,j))/
     &            (xs(ntot(j)+2,1,j) + xs(ntot(j)+2,4,j))
      else
        write (6,*) '***** MB ERROR:  Cross Section Tallies Not Correct'
      endif
   88 continue
      close (11)
      endif
c
c...Modify library
c
      totpwr = 0.0
      totfis = 0.0
      if (posit.eq.'m') then
      do 260 j=1,nmat
```

```
      mtu(j) = 0.0
      write (6,*) '...MB: Modifying Library for material ',j
      if (j.lt.10) then
        fort7 = 'fort_'//char(j+48)//'.7'
        fort9 = 'fort_'//char(j+48)//'.9'
        f9tmp = 'fort_'//char(j+48)//'.9.tmp'
        mtuf  = './tmpfile/mtu_'//char(j+48)//'.tmp'
      elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        fort7 = 'fort_'//char(j1+48)//char(j2+48)//'.7'
        fort9 = 'fort_'//char(j1+48)//char(j2+48)//'.9'
        f9tmp = 'fort_'//char(j1+48)//char(j2+48)//'.9.tmp'
        mtuf  = './tmpfile/mtu_'//char(j1+48)//char(j2+48)//'.tmp'
      endif
      open (12,file=fort9,status='old')
      open (13,file=f9tmp,status='unknown')
      if (nrst.eq.0) open (17,file=mtuf,status='unknown')
c
   90 ixs=0
      read (12,913,err=97,end=99) nflag,blanks
      if (nflag.gt.3.and.blanks.ne.'    ') then
        backspace(12)
   92   read (12,921,err=92)
     &  nxs,nnuc,xs1,xs2,xs3,xs4,xs5,xs6,xflag
        do 95 i=1,ntot(j)
        if (nisnr(i,j).eq.nnuc) then
         ixs=1
         write (13,921) nxs,nnuc,(xs(i,m,j),m=1,4),xs5,xs6,xflag
        end if
   95   continue
      end if
   97 if (ixs.eq.0) then
        backspace (12)
        read (12,'(a80)') ju80
        write (13,'(a80)') ju80
      end if
      goto 90
  913 format (i4,a4)
  921 format (i4,i8,1p6e10.3,f7.1)
c
   99 continue
      close (12)
      close (13)
c
c...Calculate energy per fission qfis and flux norm factor
c...need to determine contribution of each iso to fission
c
  100 qrat=1.0
      if (qu235.lt.0.) call calcq(qrat,fort7,fort9)
      qfis(j)=abs(qu235)*qrat
```

```
c
c...Calculate the macroscopic fission cross section of the
c...isotopes from the number densities multiplied by the
c...microscopic fission cross section
c
c...Read fort.7 and fort.9 to get density and fis xs
c
      open (16,file=fort7,status='old')
      open (13,file=f9tmp,status='old')
      nact = 27
c
c...Calc relative fission per nuclide
c
      fismac(j) = 0.
      n2nmac(j) = 0.
      absmac(j) = 0.
      n = 0
  220 read (16,911,err=220,end=250) kxs,(nisq(m),gad(m),m=1,4)
      do 240 m=1,4
       ixs=0
  230  read (13,913,err=235,end=239) nflag,blanks
       if (nflag.gt.3.and.blanks.ne.'    ') then
        backspace(13)
  232   read (13,921,err=232)
     &  nxs,nnuc,xs1,xs2,xs3,xs4,xs5,xs6,xflag
       else
        goto 230
       endif
       if (nnuc.eq.nisq(m)) ixs=1
  235  if (ixs.eq.0) goto 230
c
       if (voli(j).eq.0.0) voli(j) = vol(j)
       aval(m,j) = gad(m)*0.6022/voli(j)
       absmac(j) = absmac(j) + aval(m,j)*xs1
       n2nmac(j) = n2nmac(j) + aval(m,j)*xs2
       if (kxs.eq.2) fismac(j) = fismac(j) + aval(m,j)*xs4
       nisq1=nisq(m)/10
       nz=nisq1/1000
       a=float(nisq1)-float(1000*(nisq1/1000))
       if (nrst.eq.0) then
         if (nz.ge.90) then
           mtu(j) = mtu(j) + gad(m)*a
         endif
       endif
       n = n + 1
c
  239 if (ixs.eq.0) rewind(13)
  240 continue
      goto 220
c
c...Two different fluxes must be calculated:  one for the end
```

```
c...of step nrst, and one for the beginning of step (nrst+1)
c...The reason these two values are different is that the
c...power fraction for each outer loop step is different
c
  250 totpwr = totpwr + (qfis(j)*flx(j)*fismac(j)*voli(j))
      totfis = totfis + (flx(j)*fismac(j)*voli(j))
      qave = totpwr/totfis
      if (nrst.eq.0) write (17,'(1pe10.3)') mtu(j)
  260 continue
c
      if (nrst.eq.0) then
        pfrac1 = pfra(1)
        pfrac2 = pfra(1)
      elseif (nrst.eq.nouter) then
        pfrac1 = pfra(nrst)
        pfrac2 = pfra(nrst)
      else
        pfrac1 = pfra(nrst)
        pfrac2 = pfra(nrst+1)
      endif
c
c... Normalize the flux obtained from MCNP by using the factors "nu"
c... power, energy per fission, and k-eff
c
      if (fsrc.eq.0.) then
        fnorm = nu*1.0e+6*pow*pfrac1/1.602e-13/qave/keff
        f2norm = nu*1.0e+6*pow*pfrac2/1.602e-13/qave/keff
      else
        fnorm = src*1.0e+6*pow*pfrac1/1.602e-13/qave/floss
        f2norm = src*1.0e+6*pow*pfrac2/1.602e-13/qave/floss
      endif
c
c... Write xs data to various mb files
c
      do 160 j=1,nmat
      if (tal.ne.'yes') goto 120
      fsabs=xs(ntot(j)+1,1,j)
      fsfis=xs(ntot(j)+1,4,j)
      fsn2n=xs(ntot(j)+1,2,j)
      falabs=xs(ntot(j)+2,1,j)
      falfis=xs(ntot(j)+2,4,j)
      faln2n=xs(ntot(j)+2,2,j)
c
      if (j.lt.10) then
        file1 = 'mb1_'//char(j+48)//'.out'
        file1t= 'mb1t_'//char(j+48)//'.out'
        file2 = 'mb2_'//char(j+48)//'.out'
        file3 = 'mb3_'//char(j+48)//'.out'
        file8 = 'mb8_'//char(j+48)//'.out'
        mtuf  = './tmpfile/mtu_'//char(j+48)//'.tmp'
      elseif (j.ge.10) then
```

```
        j1 = j/10
        j2 = j - j1*10
        file1 = 'mb1_'//char(j1+48)//char(j2+48)//'.out'
        file1t= 'mb1t_'//char(j1+48)//char(j2+48)//'.out'
        file2 = 'mb2_'//char(j1+48)//char(j2+48)//'.out'
        file3 = 'mb3_'//char(j1+48)//char(j2+48)//'.out'
        file8 = 'mb8_'//char(j1+48)//char(j2+48)//'.out'
        mtuf  = './tmpfile/mtu_'//char(j1+48)//char(j2+48)//'.tmp'
      endif
      open (14,file=file2,status='unknown')
      write (14,'(i2,1x,1pe9.2,30e10.2)') nrst,(xs(i,1,j),i=1,nauto(j))
      close (14)
      open (14,file=file3,status='unknown')
      write (14,'(i2,1x,1pe9.2,30e10.2)') nrst,(xs(i,4,j),i=1,nauto(j))
      close (14)
      do 119 i=1,nauto(j)
      if (xs(i,1,j).ne.0.0.and.nisn(i,j).ge.89000) then
        fiscap(i) = (xs(i,4,j)/xs(i,1,j))
      else
        fiscap(i) = 0.0
      endif
  119 continue
      open (14,file=file8,status='unknown')
      write (14,'(i2,1x,0pf9.4,30f10.4)')
     & nrst,(fiscap(i),i=1,nauto(j))
      close (14)
c
c... Write mcnp output to mb1t.out
c
  120 flux(j)=fnorm*flx(j)
      flux2(j)=f2norm*flx(j)
      pwr(j)=qave*flux(j)*fismac(j)*voli(j)*1.602e-13/1.0e+6
c
c.. Calculate total accumulated burnup
c
      open (14,file=file1t,status='unknown')
      read (14,*)
      read (14,*)
      read (14,*)
      do 121 i=0,nrst-1
  121 read (14,'(43x,0pf10.3)') burnup(i,j)
      close (14)
      if (nrst.ge.1) then
        open (17,file=mtuf,status='unknown')
        read (17,'(1pe10.3)') mtu(j)
      endif
      if (mtu(j).ne.0.0.and.nrst.ne.0) then
       burnup(nrst,j) = burnup(nrst-1,j)
     &                + pwr(j)*1000.0*day(nrst)/mtu(j)
      else
       burnup(nrst,j) = 0.0
```

```
      endif
      write (6,900) j,flux(j),fismac(j),pwr(j),burnup(nrst,j)
c
c
      if (fsfis.ne.0.0.and.fsabs.ne.0.0) then
        fisabs = fsfis/fsabs
      else
        fisabs = 0.0
      endif
      if ((nu*fsfis+2.*fsn2n).ne.0.0.and.(fsabs+fsfis).ne.0.0)then
        eta = (nu*fsfis+2.*fsn2n)/(fsabs+fsfis)
      else
        eta = 0.0
      endif
      if (falfis.ne.0.0.and.falabs.ne.0.0) then
        fisall = falfis/falabs
      else
        fisall = 0.0
      endif
      if ((nu*falfis+2.*faln2n).ne.0.0.and.(falabs+falfis).ne.0.0)then
        aeta = (nu*falfis+2.*faln2n)/(falabs+falfis)
      else
        aeta = 0.0
      endif
      open (14,file=file1,status='unknown')
      write (14,902) nrst,qfis(j),flux(j),fismac(j),pwr(j),
     & burnup(nrst,j),
     & falabs,falfis,fisall,faln2n,aeta,fsabs,fsfis,fisabs,fsn2n,eta
      close (14)
c
c...Modify flux in origen files
c
      do 150 ii=1,2
      if (ii.eq.1) then
       if (j.lt.10) then
        fname='mbori_'//char(j+48)
        fname2='mbori_'//char(j+48)//'.tmp'
       elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        fname='mbori_'//char(j1+48)//char(j2+48)
        fname2='mbori_'//char(j1+48)//char(j2+48)//'.tmp'
       endif
      else
       i=nrst+1
       if (j.lt.10) then
        if (i.lt.10) then
         fname='./tmpfile/mbori_'//char(j+48)//'.'//char(i+48)
         fname2='./tmpfile/mbori_'//char(j+48)//'.'//char(i+48)//'.tmp'
        else
         i1 = i/10
```

```
          i2 = i - i1*10
          fname='./tmpfile/mbori_'//char(j+48)//'.'
     &           //char(i1+48)//char(i2+48)
          fname2='./tmpfile/mbori_'//char(j+48)//'.'
     &            //char(i1+48)//char(i2+48)//'.tmp'
        end if
       elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        if (i.lt.10) then
         fname='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &           //char(i+48)
         fname2='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &            //char(i+48)//'.tmp'
        elseif (i.ge.10) then
         i1 = i/10
         i2 = i - i1*10
         fname='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &           //char(i1+48)//char(i2+48)
         fname2='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &            //char(i1+48)//char(i2+48)//'.tmp'
        end if
       endif
      end if
c
      open (12,file=fname,status='old',err=140)
      open (13,file=fname2,status='unknown')
      if (mt(j).lt.0) then
        flux(j) = 0.0
        flux2(j) = 0.0
      endif
c
  130 read (12,'(a3)',end=140) ju3
      if (ju3.eq.'IRF') then
       backspace(12)
       read (12,'(a6,1pe13.5)',end=140) ju6,dstep
       if (ii.eq.1) then
         write (13,992) dstep,flux(j)
       else
         write (13,992) dstep,flux2(j)
       endif
      else
       backspace(12)
       read (12,'(a80)',end=140) ju80
       write (13,'(a80)') ju80
      end if
      goto 130
  140 continue
      close (12)
      close (13)
  150 continue
```

```
  160 continue
c
c... Obtain power fraction for ALL steps for flux calculations
c
       if (npre.eq.0) then
       if (nfd.eq.1) then
        open (15,file='feed',status='old')
        read (15,*)
        read (15,*)
        do 111 i=1,nouter
         do 111 j=1,nmat
          if (j.eq.1) then
           read (15,*) tmst(i),day(i),pfra(i),
     &      nmt(1),nfeed(i,1),gf1(i,1),gf2(i,1),nf1(i,1),rf(i,1)
          elseif (j.ge.2) then
           read (15,*)
     &      nmt(j),nfeed(i,j),gf1(i,j),gf2(i,j),nf1(i,j),rf(i,j)
          endif
  111     continue
          close (15)
       else
         do 112 i=1,nouter
  112      pfra(i) = 1.0
       endif
c
c...Modify flux in origen files.  For zero predictor steps, modify all
fluxes
c
      do 170 j=1,nmat
      do 168 i=2,nouter
       if (j.lt.10) then
        if (i.lt.10) then
         fname='./tmpfile/mbori_'//char(j+48)//'.'//char(i+48)
         fname2='./tmpfile/mbori_'//char(j+48)//'.'//char(i+48)//'.tmp'
        else
         i1 = i/10
         i2 = i - i1*10
         fname='./tmpfile/mbori_'//char(j+48)//'.'
     &          //char(i1+48)//char(i2+48)
         fname2='./tmpfile/mbori_'//char(j+48)//'.'
     &           //char(i1+48)//char(i2+48)//'.tmp'
        end if
       elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        if (i.lt.10) then
         fname='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &          //char(i+48)
         fname2='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &           //char(i+48)//'.tmp'
        elseif (i.ge.10) then
```

```
          i1 = i/10
          i2 = i - i1*10
          fname='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &           //char(i1+48)//char(i2+48)
          fname2='./tmpfile/mbori_'//char(j1+48)//char(j2+48)//'.'
     &            //char(i1+48)//char(i2+48)//'.tmp'
         end if
        endif
c
c
c... Normalize the flux obtained from MCNP by using the factors "nu"
c... power, energy per fission, and k-eff
c
       if (fsrc.eq.0.) then
        fnrm = nu*1.0e+6*pow*pfra(i)/1.602e-13/qave/keff
        f2nrm = nu*1.0e+6*pow*pfra(i)/1.602e-13/qave/keff
       else
        fnrm = src*1.0e+6*pow*pfra(i)/1.602e-13/qave/floss
        f2nrm = src*1.0e+6*pow*pfra(i)/1.602e-13/qave/floss
       endif
       fluxy(j)=fnrm*flx(j)
       fluxy2(j)=f2nrm*flx(j)
       if (mt(j).lt.0) then
         fluxy(j) = 0.0
         fluxy2(j) = 0.0
       endif
       open (12,file=fname,status='old',err=166)
       open (13,file=fname2,status='unknown')
c
  164 read (12,'(a3)',end=166) ju3

       if (ju3.eq.'IRF') then
        backspace(12)
        read (12,'(a6,1pe13.5)',end=166) ju6,dstep
        if (ii.eq.1) then
          write (13,992) dstep,fluxy(j)
        else
          write (13,992) dstep,fluxy2(j)
        endif
       else
        backspace(12)
        read (12,'(a80)',end=166) ju80
        write (13,'(a80)') ju80
       end if
       goto 164
  166 continue
       close (12)
       close (13)
  168 continue
  170 continue
       endif
```

```
      endif
c
      open (15,file='mb11.out',status='unknown')
      if (nrst.eq.0) then
         time = 0.0
      else
       if (posit.eq.'b') then
         time = daynum - day(nrst) + 0.01
       elseif (posit.eq.'m') then
         time = daynum - day(nrst)/2.0
       else
         time = daynum
       endif
      endif
c
c...Calculate k infinity and output results
c
      if (posit.eq.'m') then
        macfis = 0.0
        macabs = 0.0
        macn2n = 0.0
        do 167 j=1,nmat
        macn2n = macn2n + n2nmac(j)
        macabs = macabs + absmac(j)
  167   macfis = macfis + fismac(j)
        kinf = (nu*macfis + 2.0*macn2n)/(macfis + macabs)
        write (15,903) nrst,posit,time,keff,relerr,nu,qave,kinf
      else
        write (15,904) nrst,posit,time,keff,relerr,nu
      endif
      close (15)
      write (6,901) keff,nu
  900 format (' ...MB: mcnp flux for material ',i3,' = ',1pe9.2,
     &        '  SigmaF = ',1pe9.2,'  power = ',0pf10.3
     &        'MW  Burnup = ',0pf10.3,' GWd/MTHM')
  901 format (' ...MB: mcnp keff = ',f7.5,'  nu = ',f5.3)
  902 format (i2,1x,0pf10.3,1p3e10.2,0pf10.3,1p4e10.2,0pf8.3,2x,
     &                      1p4e10.2,0pf8.3)
  903 format (i2,a1,1x,f8.2,1x,2f10.4,f10.3,1x,2f10.3)
  904 format (i2,a1,1x,f8.2,1x,2f10.4,2f10.3)
  911 format (i4,4(1x,i6,2x,1pe10.4))
  992 format ('IRF   ',1p2e13.5,'    2  3  4  1')
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...CALCQ calculates the MeV per fission based on fission distribution
c...and qu235 (recov. MeV per U235 fission)
c
      subroutine calcq(qrat,fort7,fort9)
```

```
c
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      character fort7*12,fort9*12,blanks*4
      dimension nisq(4),gad(4)
      dimension nisact(0:50),qract(0:50),fis(0:50)
c
      data (nisact(i),qract(i),i=0,31) /
     & 0,1.0,
     & 90227,0.9043, 90229,0.9247,
     & 90232,0.9573, 91231,0.9471,
     & 91233,0.9850, 92232,0.9553,
     & 92233,0.9881, 92234,0.9774,
     & 92235,1.0000, 92236,0.9973,
     & 92237,1.0074, 92238,1.0175,
     & 93237,1.0073, 93238,1.0175,
     & 94238,1.0175, 94239,1.0435,
     & 94240,1.0379, 94241,1.0536,
     & 94242,1.0583, 95241,1.0513,
     & 95242,1.0609, 95243,1.0685,
     & 96242,1.0583, 96243,1.0685,
     & 96244,1.0787, 96245,1.0889,
     & 96246,1.0991, 96248,1.1195,
     & 96249,1.1296, 98251,1.1501,
     & 99254,1.1807   /
      qrat=0.
      nact=31
c
c...Read fort.7 and fort.9 to get density and fis xs
c
      open (12,file=fort7,status='old')
      open (13,file=fort9,status='old')
c
c...Calc relative fission per nuclide
c
      do 10 k=0,nact
   10 fis(k)=0.
      fistot=0.
   20 read (12,911,err=20,end=50) kxs,(nisq(j),gad(j),j=1,4)
      if (kxs.eq.2) then
      do 40 j=1,4
       ixs=0
   30  read (13,913,err=35,end=39) nflag,blanks
       if (nflag.gt.3.and.blanks.ne.'    ') then
        backspace(13)
   32   read (13,921,err=32)
     &  nxs,nnuc,xs1,xs2,xs3,xs4,xs5,xs6,xflag
       endif
```

```
      if (nnuc.eq.nisq(j)) ixs=1
   35 if (ixs.eq.0) goto 30
c
      nisq1=nisq(j)/10
      kk=0
      do 37 k=1,nact
      if (nisact(k).eq.nisq1) kk=k
   37 continue
      fis(kk)=fis(kk)+gad(j)*xs4
      fistot=fistot+gad(j)*xs4
c
   39 if (ixs.eq.0) rewind(13)
   40 continue
      end if
      goto 20
c
   50 continue
c
c...Calculate Q based on fission percentage
c
      if (fistot.eq.0.) then
       qrat = 0.
      else
       do 60 k=0,nact
        qrat = qrat + fis(k)/fistot*qract(k)
   60  continue
      end if
c
  911 format (i4,4(1x,i6,2x,1pe10.4))
  913 format (i4,a4)
  921 format (i4,i8,1p6e10.3,f7.1)
c
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...WMCINP modifies the mcnp input file with new compositions, materials
c...are added if they are deemed "important players".  Data is
c...read from fort.7 in gram-atoms, and put into mass fractions.
c
      subroutine wmcinp
c
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      dimension nisq(4),gad(4),nele(4),nisop(4),gmnat(999),gmcnp(999),
     &          gden(49)
      integer o,b(10),e(10)
```

```
      double precision gm(999,49)
      character ninat*10,f7name*12,f9name*12,file7*12,file4*12,
     &          fnat*12,finp*12,fmcnp*12,nmcnp*10,blanks*4,
     &          nPufp*10,nUfp*10,f9err*8,line80*80,char5*5,
     &          line*80
c
c...Read fort.7, and fort.9.  sum total gamma and fission, and
c...then step back through and determine contributors, sum mass
c...of each contibutor.
c
      do 180 j=1,nmat
      gfp = 0.
      iflag = 0
      iflg = 0
      if (j.lt.10) then
        f7name = 'fort_'//char(j+48)//'.7'
        f9name = 'fort_'//char(j+48)//'.9'
        f9err = 'f9err_'//char(j+48)
        file4 = 'mb4_'//char(j+48)//'.out'
        file7 = 'mb7_'//char(j+48)//'.out'
        fnat = 'mnat_'//char(j+48)//'.tmp'
        fmcnp = 'mcnp_'//char(j+48)//'.inp'
        finp = 'mat_'//char(j+48)//'.inp'
      elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        f7name = 'fort_'//char(j1+48)//char(j2+48)//'.7'
        f9name = 'fort_'//char(j1+48)//char(j2+48)//'.9'
        f9err = 'f9err_'//char(j1+48)//char(j2+48)
        file4 = 'mb4_'//char(j1+48)//char(j2+48)//'.out'
        file7 = 'mb7_'//char(j1+48)//char(j2+48)//'.out'
        fnat = 'mnat_'//char(j1+48)//char(j2+48)//'.tmp'
        fmcnp = 'mcnp_'//char(j1+48)//char(j2+48)//'.inp'
        finp = 'mat_'//char(j1+48)//char(j2+48)//'.inp'
      endif
      open (12,file=f7name,status='old')
      open (13,file=f9name,status='old')
      open (15,file=f9err,status='unknown')
c
c...Sum total density, gamma and fission
c
      tden=0.
      tmas=0.
      tabs=0.
      tfis=0.
      do 15 n=1,999
      gmcnp(n) = 0.
   15 gmnat(n) = 0.
   20 read (12,911,err=20,end=50) kxs,(nisq(k),gad(k),k=1,4)
      do 40 k=1,4
      if (nisq(k).gt.0) then
```

```
      ixs=0
   30 read (13,913,err=35,end=39) nflag,blanks
      if (nflag.gt.3.and.blanks.ne.'    ') then
       backspace(13)
   32  read (13,921,err=32)
     & nxs,nnuc,xs1,xs2,xs3,xs4,xs5,xs6,xflag
      else
       goto 30
      endif
      if (nnuc.eq.nisq(k)) ixs=1
   35 if (ixs.eq.0) goto 30
   39 if (ixs.eq.0) then
        rewind(13)
        xs1 = 0.0
        xs4 = 0.0
      endif
c
      nisq1=nisq(k)/10
      a=float(nisq1)-float(1000*(nisq1/1000))
      tmas=tmas+gad(k)*a
      tden=tden+gad(k)
      tabs=tabs+gad(k)*xs1
      if (kxs.eq.2) then
        iflg = 1
        tfis=tfis+gad(k)*xs4
      endif
c
c... Obtain composition (in grams) of all isotopes in MCNP input file
c...    to transfer them in case they are not found "important"
c
      if (kxs.eq.1.or.kxs.eq.2) then
       open (17,file=fmcnp,status='unknown')
       id = 0
       m = 0
   36  read (17,'(i5)',err=37,end=38) numcnp
       m = m + 1
        if (numcnp.eq.nisq1) then
         id = 1
         gmcnp(m)= a*gad(k)
        endif
   37  if (id.eq.0) goto 36
   38  close (17)
      elseif (kxs.eq.3) then
        gfp = gfp + a*gad(k)
      endif
      end if
   40 continue
c
c... Add up gram totals for natural isotopes
c
      backspace (12)
```

```
      read (12,912,end=49) kxs,(nele(k),nisop(k),gad(k),k=1,4)
      if (kxs.eq.1.or.kxs.eq.2) then
      do 47 k=1,4
      n = 0
      open (11,file=fnat,status='unknown')
   46 read (11,'(i2,4x,a10)',err=46,end=48) nelem,ninat
      n = n + 1
      if (nele(k).eq.nelem) then
        nisq1=nisq(k)/10
        a=float(nisq1)-float(1000*(nisq1/1000))
        gmnat(n)=gmnat(n)+a*gad(k)
      endif
      goto 46
   48 close (11)
   47 continue
      endif
   49 goto 20
c
   50 continue
      close (11)
      close (17)
  911 format (i4,4(1x,i6,2x,1pe10.4))
  912 format (i4,4(1x,i2,i4,2x,1pe10.4))
  913 format (i4,a4)
  921 format (i4,i8,1p6e10.3,f7.1)
c
c...Begin list of mcnp input isos with automatic tallies list
c
      ntot(j)=nauto(j)
c
c...Now determine which iso's contribute based on frimp or are
c..already selected (auto due to input or may occur twice in table)
c
      rewind(12)
      rewind(13)
      gmtot=0.
      U235f=0.
      Pu239f=0.
      open (16,file=file7,status='unknown')
      write (16,*)
   60 read (12,911,err=60,end=90) kxs,(nisq(k),gad(k),k=1,4)
      backspace (12)
      read (12,912) kxs,(nele(k),nisop(k),gad(k),k=1,4)
c
      do 80 k=1,4
      if (nisq(k).gt.0) then
       ixs=0
   70  read (13,913,err=75,end=79) nflag,blanks
       if (nflag.gt.3.and.blanks.ne.'    ') then
       backspace(13)
   72  read (13,921,err=72)
```

```
     &  nxs,nnuc,xs1,xs2,xs3,xs4,xs5,xs6,xflag
        else
         goto 70
        endif
        if (nnuc.eq.nisq(k)) ixs=1
   75   if (ixs.eq.0) goto 70
   79   if (ixs.eq.0)  then
          rewind(13)
          if (kxs.ne.3) then
            write (15,'(a27,i6,a20)') '***** MB WARNING:  Isotope ',
     &           nisq(k),' not found in fort.9'
          endif
          xs1 = 0.0
          xs4 = 0.0
        endif
c
c...Determine which isos qualify, or are automatic or repeat.
c
        icon=0
        nisq1=nisq(k)/10
        a=float(nisq1)-float(1000*(nisq1/1000))
        gmtot=gmtot+a*gad(k)
        gpct=gad(k)*a/tmas
        dpct=gad(k)/tden
        apct=gad(k)*xs1/tabs
        fpct=0.
        nz = nisq1/1000
        if (kxs.eq.2.and.tfis.ne.0.) then
          fpct=gad(k)*xs4/tfis
          if (nz.le.92) U235f = U235f + fpct
          if (nz.gt.92) Pu239f = Pu239f + fpct
        endif
c
        if (gpct.gt.abs(frimp)) icon=1
        if (dpct.gt.abs(frimp)) icon=1
        if (apct.gt.abs(frimp)) icon=1
        if (fpct.gt.abs(frimp)) icon=1
        kk=0
        do 77 m=1,ntot(j)
          if (nisnr(m,j).eq.nisq(k)) then
             kk=m
c
c... If a fission product is flagged "automatic", then don't include it
in
c... lump sum of FPs.   Otherwise, do. (kk=0 indicates it was not
"automatic")
c
             if (kxs.eq.3) gfp = gfp - a*gad(k)
          endif
   77   continue
c
```

```
c... Make sure natural isotopes are not deemed "important" since they
are
c...   included later
c
      open (11,file=fnat,status='unknown')
   78 read (11,'(i2,4x,a10)',err=78,end=92) nelem,ninat
      if (nele(k).eq.nelem) then
        icon = 0
      endif
      goto 78
   92 close (11)
c
c...If repeat or automatic isotope
c
       if (kk.gt.0) then
         gm(kk,j)=gm(kk,j)+a*gad(k)
         if (gm(kk,j).gt.a*gad(k)) then
         write (6,953) nrst,kk,nisnr(kk,j),gm(kk,j),gpct,dpct,apct,fpct
         write (16,953) nrst,kk,nisnr(kk,j),gm(kk,j),gpct,dpct,apct,fpct
         else
         if (icon.eq.1) write(6,951) nrst,kk,nisnr(kk,j),gm(kk,j),gpct,
     &         dpct,apct,fpct
         if (icon.eq.0) write(6,952) nrst,kk,nisnr(kk,j),gm(kk,j),gpct,
     &         dpct,apct,fpct
         if (icon.eq.1) write(16,951) nrst,kk,nisnr(kk,j),gm(kk,j),gpct,
     &         dpct,apct,fpct
         if (icon.eq.0) write(16,952) nrst,kk,nisnr(kk,j),gm(kk,j),gpct,
     &         dpct,apct,fpct
         end if
       end if
c
c... Fission products that were not previously deemed "important" will
c...   be treated as a lump sum
c
       if (kxs.eq.3.and.kk.eq.0.and.frimp.lt.0.0) then
       else
c
c... If new qualifying isotope, first check if xs exists then add to
ntot
c
       if (icon.eq.1.and.kk.eq.0) then

        open (15,file='mbxs.inp',status='unknown')
        ifd=0
   95   read (15,*,end=105) nixs
        nixs10 = nixs*10
        if (nixs.eq.95242) nixs10 = nixs10 + 1
        if (nisq(k).eq.nixs10) ifd=1
        if (ifd.eq.0) goto 95
        backspace (15)
        read (15,'(a10)') niso(ntot(j)+1,j)
```

```
c
c... Print error message if no cross section exists in MCNP for isotope
c
  105    if (ifd.eq.0) then
          write (6,*) '***** MB WARNING: mcnp xs not found  ', nisq(k)
          write (16,*) '***** MB WARNING: mcnp xs not found  ', nisq(k)
         end if
         close (15)
c
c... Print isotope-specific information if xs does exist
c
         if (ifd.eq.1) then
          ntot(j)=ntot(j)+1
          nisnr(ntot(j),j)=nisq(k)
          nisn(ntot(j),j)=nisnr(ntot(j),j)/10
          gm(ntot(j),j)=a*gad(k)
          write (6,951) nrst,ntot(j),nisnr(ntot(j),j),gm(ntot(j),j),
     &                  gpct,dpct,apct,fpct
          write (16,951) nrst,ntot(j),nisnr(ntot(j),j),gm(ntot(j),j),
     &                  gpct,dpct,apct,fpct
          end if
         end if
        endif
       end if
   80 continue
       goto 60
c
   90 continue
  951 format (i4,i4,i10,1p5e10.2)
  952 format (i4,i4,i10,1p5e10.2,'automatic')
  953 format (i4,i4,i10,1p5e10.2,'repeat')
       close (16)
c
c...Write grams of material
c
       if (posit.eq.'m') then
       open (14,file=file4,status='unknown')
       write (14,'(i2,1x,1pe9.2,30e10.2)') nrst,(gm(i,j),i=1,nauto(j))
       close (14)
       endif
c
       close (12)
       close (13)
c
c...Rewrite mb.inp
c
       call wmbinp
c
c...Check if mass of isos sent back to mcnp is same as came in.
c
       gmtot2=0.
```

```
      do 140 i=1,ntot(j)
  140 gmtot2=gmtot2+gm(i,j)
c
c...Read natural iso file and add to total mass
c
      n = 0
      open (11,file=fnat,status='unknown')
  142 read (11,'(6x,a10)',end=144) ninat
      n = n + 1
      gmtot2=gmtot2+gmnat(n)
      goto 142
  144 continue
c
c... Add isotopes in original MCNP input file
c
      m = 0
      open (17,file=fmcnp,status='unknown')
  145 read (17,'(i5,2x,a10)',end=148) nmc,nmcnp
      m = m + 1
      ifg = 0
      do 147 i=1,ntot(j)
       if (nisn(i,j).eq.nmc) ifg = 1
  147 continue
      if (ifg.ne.1) then
       gmtot2=gmtot2+gmcnp(m)
      endif
      goto 145
  148 continue
c
c... Add fission products to gram total, then separate into U-235 & Pu-
239 ones
c
      if (gfp.gt.0.0.and.frimp.lt.0.0) then
       gmtot2=gmtot2 + gfp
       gUff = U235f*gfp
       gPuff = Pu239f*gfp
      endif
c
c... Compare total of isotopes to total included in MCNP input file
c... Calculate gram density of material
c
      write (6,*) 'mass not accounted for and % ',gmtot-gmtot2,
     & (gmtot-gmtot2)/gmtot
      gden(j) = -gmtot2/voli(j)
c
c...Modify mt card with input file mat.inp
c
  160 open (12,file=finp,status='unknown')
      if (abs(mt(j)).lt.10) write (12,931) abs(mt(j))
      if (abs(mt(j)).ge.10.and.abs(mt(j)).lt.100)
     &             write (12,932) abs(mt(j))
```

```
       if (abs(mt(j)).ge.100.and.abs(mt(j)).lt.1000)
      &             write (12,933) abs(mt(j))
       if (abs(mt(j)).ge.1000.and.abs(mt(j)).lt.10000)
      &             write (12,934) abs(mt(j))
  931 format ('c'/'m',i1)
  932 format ('c'/'m',i2)
  933 format ('c'/'m',i3)
  934 format ('c'/'m',i4)
c
c... Add isotopes in original MCNP input file
c
      m = 0
      rewind (17)
  155 read (17,'(i5,2x,a10)',end=168) nmc,nmcnp
      m = m + 1
      ifg = 0
      do 157 i=1,ntot(j)
       if (nisn(i,j).eq.nmc) ifg = 1
  157 continue
      if (ifg.ne.1) then
        if (gmcnp(m).eq.0.) gmcnp(m)=1.0e-20*gmtot2
        write (12,'(6x,a10,1pe13.4)') nmcnp,-gmcnp(m)/gmtot2
      endif
      goto 155
c
c...Add natural isos
c
  168 n = 1
      rewind (11)
  152 read (11,'(i2,4x,a10)',end=154) nelem,ninat
      do 153 i=1,ntot(j)
       aa=(nisn(i,j)-1000*(nisn(i,j)/1000))
       if (aa.eq.0) then
          nz=nisn(i,j)/1000
          if (nz.eq.nelem) then
            ifg = 1
            gm(i,j) = gm(i,j) + gmnat(n)
          endif
       endif
  153 continue
      if (ifg.ne.1) then
        if (gmnat(n).eq.0.) gmnat(n)=1.0e-20*gmtot2
        write (12,'(6x,a10,1pe13.4)') ninat,-gmnat(n)/gmtot2
        n = n + 1
      endif
      goto 152
  154 n = n - 1
c
c... Add "important" isotopes
c
      do 150 i=1,ntot(j)
```

```
      if (gm(i,j).eq.0.) gm(i,j)=1.0e-20*gmtot2
      if (nisn(i,j).ne.45117.and.nisn(i,j).ne.46119) then
          write (12,'(6x,a10,1pe13.4)') niso(i,j),-gm(i,j)/gmtot2
      endif
  150 continue
c
c... Add fission products to mat.inp files
c
       if (gfp.gt.0.0.and.frimp.lt.0.0) then
       open (18,file='mbxs.inp',status='unknown')
       if (gUff.ne.0) then
        ifd=0
  158   read (18,*,end=159) nixs
        if (nixs.eq.45117) ifd = 1
        if (ifd.eq.0) goto 158
        backspace (18)
        read (18,'(a10)') nUfp
  159   if (ifd.eq.0) then
          write (6,*) '***** MB WARNING: No Uranium Fission Product ',
     &               'library was provided in mbxs.inp'
        else
          write (12,'(6x,a10,1pe13.4)') nUfp,-gUff/gmtot2
        endif
        rewind (18)
       endif
c
       if (gPuff.ne.0) then
        ifd=0
  161   read (18,*,end=162) nixs
        if (nixs.eq.46119) ifd = 1
        if (ifd.eq.0) goto 161
        backspace (18)
        read (18,'(a10)') nPufp
  162   if (ifd.eq.0) then
          write (6,*) '***** MB WARNING: No Plutonium Fission Product ',
     &              'library was provided in mbxs.inp'
        else
          write (12,'(6x,a10,1pe13.4)') nPufp,-gPuff/gmtot2
        endif
        close (18)
       endif
      endif
c
c... End main material input section
c
      write (12,'(a1)') 'c'

c
c...Write actinide tally material
c
      ii = 900 + j
```

```
      do 165 i=1,ntot(j)
  165 if (nisn(i,j).ge.89000) iflag = 1
      if (iflag.eq.1) then
       write (12,'(a1,i3)') 'm',ii
       do 170 i=1,ntot(j)
       if (nisn(i,j).ge.89000) then
       if (gm(i,j).eq.0.) gm(i,j)=1.0e-10*gmtot2
       write (12,'(6x,a10,1pe13.4)') niso(i,j),-gm(i,j)/gmtot2
       end if
  170  continue
       write (12,'(a1)') 'c'
      endif
c
      close (11)
      close (12)
      close (15)
      close (17)
  180 continue
c
c
c... Rewrite density(s) in MCNP input file
c
      nflag = 0
      open (15,file='mbmc.skl',status='unknown')
      open (17,file='mbmc.tmp',status='unknown')
  181 read (15,'(a5)',end=190) char5
      if (char5(1:1).eq.'C'.or.char5(1:1).eq.'c'.or.
     &    char5.eq.'     '.or.nflag.eq.1) then
        backspace (15)
        read (15,'(a80)') line80
        if (line80(1:42).eq.'                                          '
     &  .and.line80(43:76).eq.'                                    ') then
           nflag = 1
           write (17,'(a80)') line80
        else
           write (17,'(a80)') line80
        endif
      else
      backspace (15)
      read (15,*,err=185,end=190) ncell,nmater
      ident = 0
      do 187 j=1,nmat
      if (nmater.eq.abs(mt(j))) then
        ident = 1
        backspace (15)
        read (15,'(a80)') line80
        o = 1
        n = 1
        ncount = 1
c
```

```
c... First find the first number (ncount allows it to always start in
same position)
c
  183     if (line80(n:n).ne.' ') goto 182
          n = n + 1
          ncount = ncount + 1
          goto 183
c
c... Then identify the location of the next two numbers relative to
blanks
c
  182     if (line80(n:n).eq.' ') then
            b(o) = n
            if (o.eq.3) goto 185
            m = n
  184       if (line80(m+1:m+1).eq.' ') then
              m = m + 1
              goto 184
            else
              e(o) = m
            endif
            o = o + 1
            n = m + 1
            goto 182
          else
            n = n + 1
            goto 182
          endif
      else
          goto 187
      endif
  187 continue
  185 if (ident.eq.1) then
c
c... Replace values before density, density, and then those after
density
c
        nident = 0
        do 188 i=b(3),80
        line(i:i) = line80(i:i)
  188   if (line80(i:i).ne.' ') nident = 1
        if (ncount.ge.2) then
          do 191 i=1,ncount-1
  191     line80(i:i) = ' '
        endif
        if (e(2).le.24) then
          do 192 i=e(2)+1,25
  192     line80(i:i) = ' '
        endif
        if (nident.eq.1) then
          write (17,'(a25,f10.5)') line80(1:25),gden(j)
```

```
          do 189 i=1,(b(3)-1)
  189     line(i:i) = ' '
          write (17,'(a80)') line
        else
          write (17,'(a25,f10.5)') line80(1:25),gden(j)
        endif
      else
        backspace (15)
        read (15,'(a80)') line80
        write (17,'(a80)') line80
      endif
      endif
      goto 181
  190 return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...GRAMS reads fort.7 and prints out grams of tracked material to mb5
c
      subroutine grams
c
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      character f7name*12,file5*12,file5x*12,file12*12,fil12x*12
      dimension nisq(4),gad(4),gm(999,49)
c
      do 40 j=1,nmat
      if (j.lt.10) then
       f7name = 'fort_'//char(j+48)//'.7'
       file5 = 'mb5_'//char(j+48)//'.out'
       file12 = 'mb12_'//char(j+48)//'.out'
       fil12x = 'mb12x_'//char(j+48)//'.out'
       file5x = 'mb5x_'//char(j+48)//'.out'
      elseif (j.ge.10) then
       j1 = j/10
       j2 = j - j1*10
       f7name = 'fort_'//char(j1+48)//char(j2+48)//'.7'
       file5 = 'mb5_'//char(j1+48)//char(j2+48)//'.out'
       file12 = 'mb12_'//char(j1+48)//char(j2+48)//'.out'
       fil12x = 'mb12x_'//char(j1+48)//char(j2+48)//'.out'
       file5x = 'mb5x_'//char(j1+48)//char(j2+48)//'.out'
      endif
      open (12,file=f7name,status='old')
c
   10 read (12,911,err=10,end=30) kxs,(nisq(m),gad(m),m=1,4)
c
      do 20 m=1,4
```

```
      kk=0
      do 15 k=1,nauto(j)
      if (nisnr(k,j).eq.nisq(m)) kk=k
   15 continue
c
      if (kk.gt.0.or.nisq(m).ge.890000) then
       nisq1=nisq(m)/10
       a=float(nisq1)-float(1000*(nisq1/1000))
       if (kk.gt.0) gm(kk,j)=gm(kk,j)+a*gad(m)
       if (nisq(m).ge.890000) gm(nauto(j)+1,j)=
     &          gm(nauto(j)+1,j)+a*gad(m)
      end if
c
   20 continue
      goto 10
c
   30 continue
c
  911 format (i4,4(1x,i6,2x,1pe10.4))
c
      if (posit.eq.'e') then
      open (14,file=file5,status='unknown')
      open (15,file=file5x,status='unknown')
      write (14,'(i2,1x,1pe9.2,30e10.2)') nrst,(gm(i,j),i=1,nauto(j)+1)
      write (15,'(i2,1x,1pe13.7,30e14.7)') nrst,(gm(i,j),i=1,nauto(j)+1)
      close (14)
      close (15)
      elseif (posit.eq.'b') then
      open (14,file=file12,status='unknown')
      open (15,file=fil12x,status='unknown')
      write (14,'(i2,1x,1pe9.2,30e10.2)') nrst,(gm(i,j),i=1,nauto(j)+1)
      write (15,'(i2,1x,1pe13.7,30e14.7)') nrst,(gm(i,j),i=1,nauto(j)+1)
      close (14)
      close (15)
      endif
c
      close (12)
   40 continue
c
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...RMHALF removes 1/2 way predictor cards in mbori
c
      subroutine rmhalf(nmat)
      character ju8*8,ju80*80,fname*12,f2name*12
c
      do 140 j=1,nmat
      if (j.lt.10) then
```

```
        fname = 'mbori_'//char(j+48)
        f2name = 'mbori_'//char(j+48)//'.tmp'
      elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        fname = 'mbori_'//char(j1+48)//char(j2+48)
        f2name = 'mbori_'//char(j1+48)//char(j2+48)//'.tmp'
      endif
      open (12,file=fname,status='old')
      open (13,file=f2name,status='unknown')
c
      ino = 0
  120 read (12,'(a8)',end=125) ju8
c
      if (ju8.eq.'RDA Firs') ino=1
      if (ino.eq.0) then
       backspace(12)
       read (12,'(a80)',end=125) ju80
       write (13,'(a80)') ju80
      end if
      if (ju8.eq.'RDA Last') ino=0
      goto 120
  125 continue
      close (12)
      close (13)
  140 continue
c
      return
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...BURNCALC calculates material burned/produced based on feed and inven
c
      subroutine burnca
c
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
      character file10*12,file9t*12,file5t*12,file5x*12,file9*12
      character file5*12,fil12a*12
      dimension tfeed(999),g1(999),g2(999),bb(999),bb2(999),day(99)
      dimension dfeed(999)
c
c...Read feed data
c
      do 100 j=1,nmat
      if (j.lt.10) then
        file9t='mb9t_'//char(j+48)//'.out'
```

```
        file5 ='mb5t_'//char(j+48)//'.out'
        file5x='mb5tx_'//char(j+48)//'.out'
        fil12a='mb12a_'//char(j+48)//'.out'
        file9 ='mb9_'//char(j+48)//'.out'
        file10='mb10t_'//char(j+48)//'.out'
      elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        file9t='mb9t_'//char(j1+48)//char(j2+48)//'.out'
        file5t='mb5t_'//char(j1+48)//char(j2+48)//'.out'
        file5x='mb5tx_'//char(j1+48)//char(j2+48)//'.out'
        fil12a='mb12a_'//char(j1+48)//char(j2+48)//'.out'
        file9 ='mb9_'//char(j1+48)//char(j2+48)//'.out'
        file10='mb10t_'//char(j1+48)//char(j2+48)//'.out'
      endif
      open (11,file=file9t,status='unknown')
      read (11,'(//)')
      do 10 i=1,nrst
   10 read (11,'(3x,f8.2,3x,1pe9.2,30e10.2)')
     &            day(i),(tfeed(m),m=1,nauto(j)+1)
      close (11)
c
c...Read inventory data
c
      open (11,file=file5x,status='unknown')
      read (11,'(//)')
      do 20 i=0,nrst
   20 read (11,'(3x,1pe13.7,30e14.7)') (g2(m),m=1,nauto(j)+1)
      close (11)
      open (11,file=fil12a,status='unknown')
      read (11,'(//)')
      do 22 i=0,nrst
   22 read (11,'(3x,1pe13.7,30e14.7)') (g1(m),m=1,nauto(j)+1)
      close (11)
c
c...Write burn data
c
      do 30 m=1,nauto(j)+1
   30 bb(m)=g2(m)-g1(m)-tfeed(m)
      open (14,file=file9,status='unknown')
      write (14,'(i2,1x,1pe9.2,30e10.2)')
     &          nrst,(bb(m),m=1,nauto(j)+1)
c
c...Write final burn data if last step
c
      if (nrst.eq.nouter) then
c
      open (11,file=file9t,status='unknown')
      read (11,'(//)')
      do 40 i=1,nrst
   40 read (11,*)
```

```
      read (11,'(3x,f8.2,3x,1pe9.2,30e10.2)')
     &         day(nrst),(tfeed(m),m=1,nauto(j)+1)
      close (11)
c
      open (11,file=file5x,status='unknown')
      read (11,'(//)')
      read (11,'(3x,1pe13.7,30e14.7)') (g1(m),m=1,nauto(j)+1)
      close (11)
c
      do 50 m=1,nauto(j)+1
   50 bb2(m)=g2(m)-g1(m)-tfeed(m)
      write (14,'(a3,1pe9.2,30e10.2)') 'tot',(bb2(m),m=1,nauto(j)+1)
      write (14,'(/,a36,a13,i3,a22,i3,a1)')
     & 'Summary of Inventory/Feed/Production',
     & ' for material',j,' (MCNP Material Number',abs(mt(j)),')'
      write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j)),'actinide'
c
      write (14,'(a3,1pe9.2,30e10.2)') 'ini',(g1(m),m=1,nauto(j)+1)
      write (14,'(a3,1pe9.2,30e10.2)') 'fin',(g2(m),m=1,nauto(j)+1)
      write (14,'(a3,1pe9.2,30e10.2)') 'fed',(tfeed(m),m=1,nauto(j)+1)
      write (14,'(a3,1pe9.2,30e10.2)') 'net',(bb2(m),m=1,nauto(j)+1)
c
      end if
c
      close (14)
c
c...Write mb10.out containing feed/burn rates
c
      if (nrst.eq.nouter) then
      open (14,file='mb10',status='unknown')
      write (14,'(/,a28)') 'Monteburns Inventory (cont.)'
      close (14)
      open (14,file=file10,status='unknown')
c
c...Read data and divide by time interval
c
      open (11,file=file9t,status='unknown')
      read (11,'(//)')
      write (14,'(/,a17,a13,i3,a22,i3,a1)') 'Feed Rate (g/day)',
     & ' for material',j,' (MCNP Material Number',abs(mt(j)),')'
      write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j)),'actinide'
      do 80 i=1,nouter
      read (11,'(3x,f8.2,3x,1pe9.2,30e10.2)')
     &            day(i),(tfeed(m),m=1,nauto(j)+1)
      open (17,file='./tmpfile/params2',status='old')
      read (17,'(i4)') nfd
      close (17)
      if (nfd.eq.1) then
       write (14,'(i2,1x,1pe9.2,30e10.2)') i,
     & (tfeed(m)/day(i),m=1,nauto(j)+1)
      else
```

```
      do 77 m=1,nauto(j)+1
   77 dfeed(m) = 0.0
      write (14,'(i2,1x,1pe9.2,30e10.2)') i,
     & (dfeed(m),m=1,nauto(j)+1)
      endif
   80 continue
      read (11,'(///)')
      write (14,'(/,a35,a13,i3,a22,i3,a1)')
     & 'Production/Destruction Rate (g/day)',
     & ' for material',j,' (MCNP Material Number',abs(mt(j)),')'
      write (14,'(3x,a9,30(1x,a9))') (niso(i,j),i=1,nauto(j)),'actinide'
      do 90 i=1,nouter-1
      read (11,'(3x,1pe9.2,30e10.2)') (bb2(m),m=1,nauto(j)+1)
   90 write (14,'(i2,1x,1pe9.2,30e10.2)') i,
     & (bb2(m)/day(i),m=1,nauto(j)+1)
      write (14,'(i2,1x,1pe9.2,30e10.2)') nouter,
     & (bb(m)/day(nouter),m=1,nauto(j)+1)
      close (11)
      write (14,*)
      close (14)
      end if
  100 continue
c
      return
      end
c
c...DISCRETE makes additions in fort.7 and mat.inp for discrete feed
c
      subroutine discr
c
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      character line*80,fort7*12,f7tmp*15,met*1
      dimension nisq(4),gad(4),gmafed(99,49),ifd6(99,49),a(99)
      dimension day(99),nfeed(99,49),gf1(99,49),gf2(99,49),mfeed(99),
     & kfeed(99),kfeed1(99,99),kfeed2(99,99),ifeed(99,99),ncount(99,49),
     & nf1(99,49),rf(99,49),pfra(99),nmt(49),ffeed(99,99),
     & nelem(99,49),tmst(99)
      dimension nisoto(99,49,99),nisop(99,49),atomfr(99,49,99),
     & iflag(99,49),imfeed(99,99),fmfeed(99,99),mmfeed(99),gfeed(99,49)
c
c... Determine if feed file exists
c
      open (17,file='./tmpfile/params2',status='old')
      read (17,'(i4)') nfd
      close (17)
      if (nfd.eq.1) then
        open (11,file='feed',status='old')
        read (11,*)
        read (11,*)
```

```
          do 10 i=1,nouter
           do 10 j=1,nmat
            if (j.eq.1) then
              read (11,*) tmst(i),day(i),
     &        pfra(i),nmt(1),nfeed(i,1),gf1(i,1),gf2(i,1),nf1(i,1),rf(i,1)
           elseif (j.ge.2) then
              read (11,*) nmt(j),
     &        nfeed(i,j),gf1(i,j),gf2(i,j),nf1(i,j),rf(i,j)
           endif
   10     continue
          read (11,'(i4)') nfs
          do 12 n=1,nfs
            read (11,'(i4)') mfeed(n)
          do 12 m=1,mfeed(n)
   12       read (11,'(i5,f9.7)') ifeed(n,m),ffeed(n,m)
            read (11,'(i4)') nrs
          do 15 n=1,nrs
            read (11,'(i4)') kfeed(n)
          do 15 k=1,kfeed(n)
   15       read (11,'(i4,i4)') kfeed1(n,k),kfeed2(n,k)
        endif
c
c...Rewrite fort.7
c
        do 100 j=1,nmat
        i = nrst
        n = nfeed(i,j)
        if (j.lt.10) then
          fort7 = 'fort_'//char(j+48)//'.7'
          f7tmp = 'fort_'//char(j+48)//'.7.tmp'
        elseif (j.ge.10) then
          j1 = j/10
          j2 = j - j1*10
          fort7 = 'fort_'//char(j1+48)//char(j2+48)//'.7'
          f7tmp = 'fort_'//char(j1+48)//char(j2+48)//'.7.tmp'
        endif
        open (12,file=fort7,status='unknown')
        if (n.eq.0) then
           goto 90
        endif
        open (13,file='fort.tmp',status='unknown')
c
c... Check to see if any feed materials are natural elements
c
        mmfeed(n) = mfeed(n)
        do 25 m=1,mfeed(n)
        iflag(m,j) = 0
        nai = ifeed(n,m)-1000*(ifeed(n,m)/1000)
        if (nai.eq.0.and.ifeed(n,m).gt.0) then
          open (16,file='natelem',status='unknown')
          read (16,*)
```

```
      read (16,*)
   18 read (16,*) nelem(m,j)
      read (16,*) nisop(m,j)
      do 20 mm=1,nisop(m,j)
   20 read (16,'(i5,3x,f10.5)',err=20,end=23)
     &       nisoto(m,j,mm),atomfr(m,j,mm)
      if (nelem(m,j).eq.ifeed(n,m)/1000) then
         iflag(m,j) = 1
         imfeed(n,m) = nisoto(m,j,1)
         fmfeed(n,m) = ffeed(n,m)*atomfr(m,j,1)
        do 22 mn=1,(nisop(m,j)-1)
          imfeed(n,mmfeed(n)+mn) = nisoto(m,j,1+mn)
   22     fmfeed(n,mmfeed(n)+mn) = ffeed(n,m)*atomfr(m,j,1+mn)
        mmfeed(n) = mmfeed(n) + (nisop(m,j)-1)
        goto 23
      else
        goto 18
      endif
   23 close (16)
     else
      imfeed(n,m) = ifeed(n,m)
      fmfeed(n,m) = ffeed(n,m)
     endif
   25 continue
c
c...Convert grams of feed to gram-atoms of feed
c
      do 28 m=1,mmfeed(n)
       ifd6(n,m) = imfeed(n,m)*10
       if (ifd6(n,m).eq.952420) ifd6(n,m)=ifd6(n,m)+1
       gfeed(m,j)=fmfeed(n,m)*gf2(i,j)*day(i)
       ai = float(imfeed(n,m))-float(1000*(imfeed(n,m)/1000))
       gmafed(m,j) = gfeed(m,j)/ai
       ncount(m,j) = 0
   28 continue
   30 read (12,901,err=45,end=50) kxs,(nisq(k),gad(k),k=1,4)
  901 format (i4,4(1x,i6,2x,1pe10.4))
      if (kxs.eq.0) goto 45
      do 40 k=1,4
       do 40 m=1,mfeed(n)
       if (nisq(k).eq.ifd6(n,m).and.kxs.le.2) then
          if (ncount(m,j).eq.0) gad(k) = gad(k) + gmafed(m,j)
          ncount(m,j) = 1
       endif
   40 continue
      write (13,901) kxs,(nisq(k),gad(k),k=1,4)
      goto 30
   45 backspace (12)
      read (12,'(a80)') line
      write (13,'(a80)') line
      goto 30
```

```
   50 close (12)
      close (13)
c
c...Write non-actinides to fort.7 that are part of discrete feed but did
not
c... previously exist
c
      open (13,file='fort.tmp',status='unknown')
      open (14,file=f7tmp,status='unknown')
      kxsold = 1
      nadd = 0
   63 read (13,'(i4)',err=80,end=99) kxs
      if (kxs.eq.kxsold) then
        backspace (13)
        read (13,'(a80)') line
        write (14,'(a80)') line
        kxsold = kxs
      else
      kxsold = kxs
      if (nadd.eq.0) then
      do 65 k=1,mmfeed(n)
       nmin=99999
       ni=0
       do 60 m=1,mmfeed(n)
        a(m)=float(imfeed(n,m))-float(1000*(imfeed(n,m)/1000))
        if (imfeed(n,m).lt.83000.and.imfeed(n,m).gt.1000) then
         if (a(m).gt.0) then
          if (imfeed(n,m).lt.nmin) then
           nmin=imfeed(n,m)
           ni=m
          end if
         endif
        end if
   60  continue
       if (ni.gt.0) then
        kxs=1
        met='0'
        if (ncount(ni,j).eq.0) then
          ncount(ni,j) = 1
          write (14,912) kxs,ifd6(n,ni),gmafed(ni,j)
        endif
        imfeed(n,ni)=0
       end if
   65  continue
c
c...Write actinides to fort.7, sort numerically for xs file read
c
      do 75 k=1,mmfeed(n)
       nmin=99999
       ni=0
       do 70 m=1,mmfeed(n)
```

```
        a(m)=float(imfeed(n,m))-float(1000*(imfeed(n,m)/1000))
        if (imfeed(n,m).ge.83000.and.a(m).gt.0.) then
         if (imfeed(n,m).lt.nmin) then
          nmin=imfeed(n,m)
          ni=m
         end if
        end if
   70  continue
       if (ni.gt.0) then
        kxs=2
        met='0'
        if (ncount(ni,j).eq.0) then
          ncount(ni,j) = 1
          write (14,912) kxs,ifd6(n,ni),gmafed(ni,j)
        endif
        imfeed(n,ni)=0
       end if
   75 continue
      nadd = 1
      endif
      if (kxsold.eq.0) goto 80
      backspace(13)
      read (13,'(a80)') line
      write (14,'(a80)') line
      endif
      goto 63
   80 backspace(13)
      read (13,'(a80)') line
      write (14,'(a80)') line
      goto 63
   90 open (14,file=f7tmp,status='unknown')
   95 read (12,'(a80)',end=99) line
      write (14,'(a80)') line
      goto 95
   99 close (13)
      close (14)
  100 continue
  911 format (i4,i6,a1,1pe12.4,
     & '      0  0.0000E+00      0  0.0000E+00      0  0.0000E+00')
  912 format (i4,1x,i6,1pe12.4,
     & '      0  0.0000E+00      0  0.0000E+00      0  0.0000E+00')
       end
c
      subroutine dremo
c
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      character line*72,fort7*12,f7tmp*15,nisq2(4)*4
      dimension nisq1(4),gad(4),nisq3(4)
      dimension day(99),nfeed(99,49),gf1(99,49),gf2(99,49),mfeed(99),
```

```
     & kfeed(99),kfeed1(99,99),kfeed2(99,99),ifeed(99,99),tmst(99),
     & nf1(99,49),rf(99,49),pfra(99),nmt(49),ffeed(99,99)
c
c... Determine if feed file exists
c
      open (17,file='./tmpfile/params2',status='old')
      read (17,'(i4)') nfd
      close (17)
      if (nfd.eq.1) then
        open (11,file='feed',status='old')
        read (11,*)
        read (11,*)
        do 10 i=1,nouter
         do 10 j=1,nmat
          if (j.eq.1) then
            read (11,*) tmst(i),day(i),pfra(i),
     &       nmt(1),nfeed(i,1),gf1(i,1),gf2(i,1),nf1(i,1),rf(i,1)
          elseif (j.ge.2) then
            read (11,*)
     &       nmt(j),nfeed(i,j),gf1(i,j),gf2(i,j),nf1(i,j),rf(i,j)
          endif
   10     continue
          read (11,'(i4)') nfs
          do 12 n=1,nfs
            read (11,'(i4)') mfeed(n)
          do 12 m=1,mfeed(n)
   12       read (11,'(i5,f9.7)') ifeed(n,m),ffeed(n,m)
            read (11,'(i4)') nrs
          do 15 n=1,nrs
            read (11,'(i4)') kfeed(n)
          do 15 k=1,kfeed(n)
   15       read (11,'(i4,i4)') kfeed1(n,k),kfeed2(n,k)
        endif
c
c...Rewrite fort.7
c
      do 60 j=1,nmat
      if (nf1(nrst,j).ge.0) goto 60
      if (j.lt.10) then
        fort7 = 'fort_'//char(j+48)//'.7'
        f7tmp = 'fort_'//char(j+48)//'.7.tem'
      elseif (j.ge.10) then
        j1 = j/10
        j2 = j - j1*10
        fort7 = 'fort_'//char(j1+48)//char(j2+48)//'.7'
        f7tmp = 'fort_'//char(j1+48)//char(j2+48)//'.7.tem'
      endif
      open (12,file=fort7,status='unknown')
      open (13,file=f7tmp,status='unknown')
c
c... Remove elements in removal group from fort.7
```

```
c
   30 read (12,901,err=45,end=50) kxs,(nisq1(k),nisq2(k),gad(k),k=1,4)
      backspace (12)
      read (12,903,err=45,end=50) kxs,(nisq3(k),nisq2(k),gad(k),k=1,4)
  901 format (i4,4(1x,i2,a4,2x,1pe10.4))
  903 format (i4,4(1x,a2,a4,2x,1pe10.4))
      nrem = abs(nf1(nrst,j))
      do 40 k=1,4
       do 40 n=1,kfeed(nrem)
        do 40 m=abs(kfeed1(nrem,n)),abs(kfeed2(nrem,n))
        if (nisq1(k).eq.m) then
          if ((kfeed1(nrem,n).lt.0.and.kxs.eq.3)
     &                              .or.kfeed1(nrem,n).ge.0) then
            gad(k) = gad(k) - gad(k)*rf(nrst,j)
          endif
        endif
   40  continue
       if (kxs.eq.0) then
         write (13,902) kxs,(nisq2(k),gad(k),k=1,4)
  902 format (i4,4(1x,2x,a4,2x,1pe10.4))
       else
         write (13,903) kxs,(nisq3(k),nisq2(k),gad(k),k=1,4)
       endif
       goto 30
   45  backspace (12)
       read (12,'(a72)') line
       write (13,'(a72)') line
       goto 30
   50 close (12)
      close (13)
   60 continue
      end

c
c...REGION makes indicates what materials are substituted in various
c... regions
c
      subroutine region
c
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      character fname*25
      dimension day(99),pfra(99),nmt(49),nfeed(99,49),gf1(99,49),
     &          gf2(99,49),nf1(99,49),rf(99,49),tmst(99)
c
c...First discover if feed input file exists
c
      open (17,file='./tmpfile/params2',status='old')
      read (17,'(i4)') nfd
      if (nfd.eq.1) then
```

```
c
c...First read the two lines of headings
c
      open (11,file='feed',status='unknown')
      read (11,*)
      read (11,*)
      do 8 i=1,nrst
        do 8 j=1,nmat
        if (j.eq.1) then
         read (11,*) tmst(i),day(i),
     &   pfra(i),nmt(1),nfeed(i,1),gf1(i,1),gf2(i,1),nf1(i,1),rf(i,1)
        elseif (j.ge.2) then
         read (11,*) nmt(j),
     &   nfeed(i,j),gf1(i,j),gf2(i,j),nf1(i,j),rf(i,j)
        endif
    8  continue
       close (11)
      else
       do 10 j=1,nmat
   10  nmt(j) = 0
      endif
c
      do 20 j=1,nmat
      if (j.lt.10) then
       fname = './tmpfile/param3_'//char(j+48)
      elseif (j.ge.10) then
       j1 = j/10
       j2 = j - j1*10
       fname = './tmpfile/param3_'//char(j1+48)//char(j2+48)
      endif
      open (12,file=fname,status='unknown')
      write (12,905) nmt(j)
  905 format (i4,'  nval')
   20 continue
      end
c
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c...WMBINP rewrites mb.inp
c
      subroutine wmbinp
c
      common /mbinp/nmat,mt(49),voli(49),pow,qu235,days,nouter,ninner,
     & npre,nrst,frimp,nauto(49),ntot(49),nkeff,nisn(999,49),
     & nisnr(999,49)
      common /mbinp2/niso(999,49),nisor(999,49),title,olib,locale,posit
      character niso*10,nisor*6,title*72,olib*2,locale*72,posit*1
c
c...Rewrite mb.inp
c
      open (11,file='mb.inp',status='unknown')
```

```
      write (11,'(a72)') title
      write (11,*) nmat
      do 20 j=1,nmat
   20 write (11,*) mt(j)
      do 30 j=1,nmat
   30 write (11,*) voli(j)
      write (11,*) pow
      write (11,*) qu235
      write (11,*) days
      write (11,*) nouter
      write (11,*) ninner
      write (11,*) npre
      write (11,*) nrst
      write (11,'(a2)') olib
      write (11,'(a72)') locale
      write (11,*) frimp
      write (11,*) nkeff
      do 60 j=1,nmat
      write (11,*) nauto(j)
      write (11,*) ntot(j)
      do 60 i=1,ntot(j)
   60 write (11,'(a10)') niso(i,j)
      close (11)
c
      return
      end
c23456789*123456789*123456789*123456789*123456789*123456789*123456789*123456789*12
c
c  This subroutine creates a file containig isotopic breakdowns
c  for natural elements
c
      subroutine natele
      dimension nelem(40),nisot(40,40),atomfr(40,40),nisop(40)
c
c  Isotopic compositions of natural elements
c  Ref:  Nuclides and Isotopes, Fifteenth Edition
c
      open (16,file='natelem',status='unknown')
      data (nelem(i),i=1,33) /
     &    6, 12, 14, 16, 17, 18, 19, 20, 22,
     &   23, 24, 25, 26, 28, 29, 30, 31, 40,
     &   42, 47, 48, 49, 50, 51, 54, 63, 64,
     &   72, 74, 77, 78, 80, 82    /
      write (16,*)
      write (16,*)
      do 80 i=1,33
      nz = nelem(i)
      if (nz.eq.6) then                  !  Carbon
        niso = 2                  !  Number of isotopes in natural carbon
        data (nisot(1,n),atomfr(1,n),n=1,2) /
     &    6012,  0.98900,
```

```
&    6013,   0.01100    /
 endif
 if (nz.eq.12) then                    !  Magnesium
   niso = 3                    !  Number of isotopes in natural carbon
   data (nisot(2,n),atomfr(2,n),n=1,3) /
&    12024,    0.78990,
&    12025,    0.10000,
&    12026,    0.11010   /
 endif
 if (nz.eq.14) then                    !  Silicon
   niso = 3                    !  Number of isotopes in natural carbon
   data (nisot(3,n),atomfr(3,n),n=1,3) /
&    14028,    0.92230,
&    14029,    0.04670,
&    14030,    0.03100   /
 endif
 if (nz.eq.16) then                    !  Sulfur
   niso = 4                    !  Number of isotopes in natural carbon
   data (nisot(4,n),atomfr(4,n),n=1,4) /
&    16032,    0.95020,
&    16033,    0.00750,
&    16034,    0.04210,
&    16036,    0.00020   /
 endif
 if (nz.eq.17) then                    !  Chlorine
   niso = 2                    !  Number of isotopes in natural carbon
   data (nisot(5,n),atomfr(5,n),n=1,2) /
&    17035,    0.75770,
&    17037,    0.24230   /
 endif
 if (nz.eq.18) then                    !  Argon
   niso = 3                    !  Number of isotopes in natural carbon
   data (nisot(6,n),atomfr(6,n),n=1,3) /
&    18036,    0.00337,
&    18038,    0.00063,
&    18040,    0.99600   /
 endif
 if (nz.eq.19) then                    !  Potassium
   niso = 3                    !  Number of isotopes in natural carbon
   data (nisot(7,n),atomfr(7,n),n=1,3) /
&    19039,    0.93258,
&    19040,    0.00012,
&    19041,    0.06730    /
 endif
 if (nz.eq.20) then                    !  Calcium
   niso = 6                    !  Number of isotopes in natural carbon
   data (nisot(8,n),atomfr(8,n),n=1,6) /
&    20040,    0.96941,
&    20042,    0.00647,
&    20043,    0.00135,
&    20044,    0.02086,
```

```
&    20046,   0.00004,
&    20048,   0.00187   /
 endif
 if (nz.eq.22) then                ! Titanium
   niso = 5                  ! Number of isotopes in natural carbon
   data (nisot(9,n),atomfr(9,n),n=1,5) /
&    22046,   0.08250,
&    22047,   0.07440,
&    22048,   0.73720,
&    22049,   0.05410,
&    22050,   0.05180   /
 endif
 if (nz.eq.23) then                ! Vanadium
   niso = 2                  ! Number of isotopes in natural carbon
   data (nisot(10,n),atomfr(10,n),n=1,2) /
&    23050,   0.00250,
&    23051,   0.99750   /
 endif
 if (nz.eq.24) then                ! Chromium
   niso = 4                  ! Number of isotopes in natural carbon
   data (nisot(11,n),atomfr(11,n),n=1,4) /
&    24050,   0.04350,
&    24052,   0.83790,
&    24053,   0.09500,
&    24054,   0.02360   /
 endif
 if (nz.eq.25) then                ! Manganese
   niso = 1                  ! Number of isotopes in natural carbon
   data (nisot(12,n),atomfr(12,n),n=1,1) /
&    25055,   0.10000   /
 endif
 if (nz.eq.26) then                ! Iron
   niso = 4                  ! Number of isotopes in natural carbon
   data (nisot(13,n),atomfr(13,n),n=1,4) /
&    26054,   0.05850,
&    26056,   0.91750,
&    26057,   0.02120,
&    26058,   0.00280   /
 endif
 if (nz.eq.28) then                ! Nickel
   niso = 5                  ! Number of isotopes in natural carbon
   data (nisot(14,n),atomfr(14,n),n=1,5) /
&    28058,   0.68080,
&    28060,   0.26220,
&    28061,   0.01140,
&    28062,   0.03630,
&    28064,   0.00930   /
 endif
 if (nz.eq.29) then                ! Copper
   niso = 2                  ! Number of isotopes in natural carbon
   data (nisot(15,n),atomfr(15,n),n=1,2) /
```

```
&    29063,   0.69170,
&    29065,   0.30830   /
 endif
 if (nz.eq.30) then                ! Zinc
   niso = 4                  ! Number of isotopes in natural carbon
   data (nisot(16,n),atomfr(16,n),n=1,4) /
&    30064,   0.48600,
&    30066,   0.27900,
&    30067,   0.04100,
&    30068,   0.18800   /
 endif
 if (nz.eq.31) then                ! Gallium
   niso = 2                  ! Number of isotopes in natural carbon
   data (nisot(17,n),atomfr(17,n),n=1,2) /
&    31069,   0.60110,
&    31071,   0.39890   /
 endif
 if (nz.eq.40) then                ! Zirconium
   niso = 5                  ! Number of isotopes in natural carbon
   data (nisot(18,n),atomfr(18,n),n=1,5) /
&    40090,   0.51450,
&    40091,   0.11220,
&    40092,   0.17150,
&    40094,   0.17380,
&    40096,   0.02800   /
 endif
 if (nz.eq.42) then                ! Molybdenum
   niso = 7                  ! Number of isotopes in natural carbon
   data (nisot(19,n),atomfr(19,n),n=1,7) /
&    42092,   0.14840,
&    42094,   0.09250,
&    42095,   0.15920,
&    42096,   0.16680,
&    42097,   0.09550,
&    42098,   0.24130,
&    42100,   0.09630   /
 endif
 if (nz.eq.47) then                ! Silver
   niso = 2                  ! Number of isotopes in natural carbon
   data (nisot(20,n),atomfr(20,n),n=1,2) /
&    47107,   0.51839,
&    47109,   0.48161   /
 endif
 if (nz.eq.48) then                ! Cadmium
   niso = 8                  ! Number of isotopes in natural carbon
   data (nisot(21,n),atomfr(21,n),n=1,8) /
&    48106,   0.01250,
&    48108,   0.00890,
&    48110,   0.12490,
&    48111,   0.12800,
&    48112,   0.24130,
```

```
     &    48113,   0.12220,
     &    48114,   0.28730,
     &    48116,   0.07490   /
      endif
      if (nz.eq.49) then                ! Indium
        niso = 2                  ! Number of isotopes in natural carbon
        data (nisot(22,n),atomfr(22,n),n=1,2) /
     &    49113,   0.04290,
     &    49115,   0.95710   /
      endif
      if (nz.eq.50) then                ! Tin
        niso = 10                    ! Number of isotopes in natural
carbon
        data (nisot(23,n),atomfr(23,n),n=1,10) /
     &    50112,   0.00970,
     &    50114,   0.00650,
     &    50115,   0.00340,
     &    50116,   0.14540,
     &    50117,   0.07680,
     &    50118,   0.24220,
     &    50119,   0.08590,
     &    50120,   0.32590,
     &    50122,   0.04630,
     &    50124,   0.05790   /
      endif
      if (nz.eq.51) then                ! Antimony
        niso = 2                  ! Number of isotopes in natural carbon
        data (nisot(24,n),atomfr(24,n),n=1,2) /
     &    51121,   0.57300,
     &    51123,   0.42700   /
      endif
      if (nz.eq.54) then                ! Xenon
        niso = 9                  ! Number of isotopes in natural carbon
        data (nisot(25,n),atomfr(25,n),n=1,9) /
     &    54124,   0.00100,
     &    54126,   0.00090,
     &    54128,   0.01910,
     &    54129,   0.26400,
     &    54130,   0.04100,
     &    54131,   0.21200,
     &    54132,   0.26900,
     &    54134,   0.10400,
     &    54136,   0.08900   /
      endif
      if (nz.eq.63) then                ! Europium
        niso = 2                  ! Number of isotopes in natural carbon
        data (nisot(26,n),atomfr(26,n),n=1,2) /
     &    63151,   0.47800,
     &    63153,   0.52200   /
      endif
      if (nz.eq.64) then                ! Gadolinium
```

```
         niso = 7                    !  Number of isotopes in natural carbon
         data (nisot(27,n),atomfr(27,n),n=1,7) /
     &     64152,    0.00200,
     &     64154,    0.02180,
     &     64155,    0.14800,
     &     64156,    0.20470,
     &     64157,    0.15650,
     &     64158,    0.24840,
     &     64160,    0.21860    /
        endif
        if (nz.eq.72) then                    !  Hafnium
         niso = 6                    !  Number of isotopes in natural carbon
         data (nisot(28,n),atomfr(28,n),n=1,6) /
     &     72174,    0.00162,
     &     72176,    0.05206,
     &     72177,    0.18606,
     &     72178,    0.27297,
     &     72179,    0.13629,
     &     72180,    0.35100    /
        endif
        if (nz.eq.74) then                    !  Tungsten
         niso = 5                    !  Number of isotopes in natural carbon
         data (nisot(29,n),atomfr(29,n),n=1,5) /
     &     74180,    0.00120,
     &     74182,    0.26498,
     &     74183,    0.14314,
     &     74184,    0.30642,
     &     74186,    0.28426    /
        endif
        if (nz.eq.77) then                    !  Iridium
         niso = 2                    !  Number of isotopes in natural carbon
         data (nisot(30,n),atomfr(30,n),n=1,2) /
     &     77191,    0.37300,
     &     77193,    0.62700    /
        endif
        if (nz.eq.78) then                    !  Platinum
         niso = 6                    !  Number of isotopes in natural carbon
         data (nisot(31,n),atomfr(31,n),n=1,6) /
     &     78190,    0.00010,
     &     78192,    0.00790,
     &     78194,    0.32900,
     &     78195,    0.33800,
     &     78196,    0.25300,
     &     78198,    0.07200    /
        endif
        if (nz.eq.80) then                    !  Mercury
         niso = 7                    !  Number of isotopes in natural
carbon
         data (nisot(32,n),atomfr(32,n),n=1,7) /
     &     80196,    0.00150,
     &     80198,    0.09970,
```

```
     &    80199,    0.16870,
     &    80200,    0.23100,
     &    80201,    0.13180,
     &    80202,    0.29860,
     &    80204,    0.06870   /
      endif
      if (nz.eq.82) then                  !  Lead
        niso = 4                  !  Number of isotopes in natural carbon
        data (nisot(33,n),atomfr(33,n),n=1,4) /
     &    82204,    0.01400,
     &    82206,    0.24100,
     &    82207,    0.22100,
     &    82208,    0.52400   /
      endif
c
      nisop(i) = niso
      write (16,*) nelem(i)
      write (16,*) nisop(i)
      do 60 n=1,nisop(i)
   60 write (16,'(i5,3x,f10.5)')
     &        nisot(i,n),atomfr(i,n)
   80 continue
      close (16)
c
      return
      end
```

APPENDIX C. SAMPLE MCNP INPUT FILE

```
MCNP Input File for Test Case #2
C  Cell Cards
C  Irradiation of a Single Pin
C  Fuel Pin
1  1  -10.045        -1   -2   3                u=2      imp:n=1 $fuel
8  8  -0.781e-3      1  -4  -2  3                u=2      imp:n=1 $gap
6  6  -6.44          4  -6  -2  3                u=2      imp:n=1 $clad
7  7  -0.7569          6                        u=2      imp:n=1 $wat
C  Pin Cell
20 0                -9 10 -11 12 -7 8         fill=2      imp:n=1
99 0                      #20                             imp:n=0

C  Fuel Rod
1  cz  0.47815
C  Axial Distribution
2  pz   347.4
3  pz   0.0
C  Gap
4  cz  0.493
C  Fuel Cladding
6  cz  0.559
C  Unit Cell (Pitch)
7  pz 347.3
8  pz 0.1
*9  px  0.7793
*10 px -0.7793
*11 py  0.7793
*12 py -0.7793

C  Control Cards
kcode  1000 1.0 15 115
ksrc  0 0 173.6
tmp   7.25e-8    6.5e-8  5.34e-8   4.81e-8   6.0e-8   6.0e-8
C  Material Cards
C  Fuel
m1   92234.88c  6.15165e-6  92235.88c 6.89220e-4  92236.88c 3.16265e-6
     92238.88c  2.17104e-2
      6000.88c      9.13357e-6      7014.88c    1.04072e-5        8016.88c
4.48178e-2
C Cladding
m6   26000.85c  -0.005   40000.65c  -0.9791    50000  -0.0159
C Coolant
m7   1001 5.06153e-2 8016.85c 2.53076e-2
      5010.85c 2.75612e-6 5011.85c 1.11890e-5
mt7 lwtr.04t
C Gap
m8  2004.85c   -1.0
```

APPENDIX D. SAMPLE *MONTEBURNS* INPUT FILE

```
Monteburns Input File for Test Case #2
2              ! Number of MCNP materials
1              ! MCNP Material #1 (must be less than 100)
-7             ! MCNP Material #2
249.378        ! Material volume #1
502.44         ! Material volume #2
0.001          ! Total Power of System (in MWt)
-200.          ! Recov. energy/fission (MeV); 0. uses default value
0.             ! Total number of days burned (used if no feed)
8              ! Number of outer burn steps
40             ! Number of internal burn steps (multiple of 10)
1              ! Number of predictor steps (+1 on first step)
0              ! Step number to restart after (0=beginning)
22             ! Number of origen2 library
/export/io1/dip/origen/libraries ! location of ORIGEN2 library
1.0            ! Importance Fraction
0              ! Intermediate keff calc. 0) No 1) Yes
28             ! Automatic Isotopes for Region 1
92234.88c
92235.88c
92236.88c
92238.88c
93237.88c
94238.88c
94239.88c
94240.88c
94241.88c
94242.88c
95241.88c
95243.88c
42095.88c
43099.88c
44101.88c
45103.88c
47109.88c
55133.88c
55135.88c
60143.88c
60145.88c
62147.88c
62149.88c
62150.88c
62151.88c
62152.88c
63153.88c
64155.88c
2              ! Automatic Isotopes for Region 2
5010.85c
5011.85c
```

APPENDIX E. SAMPLE FEED INPUT FILE

```
Step    Time    PowFr.mat# Feed  Beg.Rate    End    Rem# Fract.Rem.
 int    real    real   int int    real       real    int  real
   1   306.0   38.066  1   0      0.0         0.0     0  0.000
                       2   0      0.0         0.0     0  0.000
   2    71.0   0.000   1   0      0.0         0.0     0  0.000 !
                       2   0      0.0         0.0    -1  1.000
   3   381.7   42.9015 1   0      0.0         0.0     0  0.000 !
                       2   1     -2.0     4.684e-4    0  0.000
   4    83.1   0.000   1   0      0.0         0.0     0  0.000 !
                       2   0      0.0         0.0    -1  1.000
   5   466.0   37.624  1   0      0.0         0.0     0  0.000 !
                       2   1     -2.0     4.118e-4    0  0.000
   6    85.0   0.000   1   0      0.0         0.0     0  0.000 !
                       2   0      0.0         0.0    -1  1.000
   7   461.1   32.171  1   0      0.0         0.0     0  0.000 !
                       2   1     -2.0     4.066e-4    0  0.000
   8  1870.0   0.000   1   0      0.0         0.0     0  0.000 !
                       2   0      0.0         0.0     0  0.000
   1                              ! # of feed specs
   2                              ! # isos in Feed #1
5010   .20                        ! B-10
5011   .80                        ! B-11
   1                              ! # of removal groups
   1                              ! # of ranges in removal group
   5   5                          ! 1st range for Feed #1 (B)
```

REFERENCES

1. J.F. BRIESMEISTER, "MCNP™ - A General Monte Carlo N-Particle Transport Code," LA-12625-M, Version 4B, Los Alamos National Laboratory (March 1997).

2. A. G. CROFF, "A User's Manual for ORIGEN2 Computer Code," ORNL/TM-7175, Oak Ridge National Laboratory (July 1980).

3. F. VENNERI, N. LI, M. A. WILLIAMSON, M. G. HOUTS, and G. P. LAWRENCE, "Disposition of Nuclear Waste Using Subcritical Accelerator-Driven Systems: Technology Choices and One Implementation Scenario," LA-UR-98-985, Los Alamos National Laboratory (March 1998).

4. J. J. DUDERSTADT and L. J. HAMILTON, *Nuclear Reactor Analysis*, pp. 76, 338, 634-635, John Wiley & Sons, New York (1976).

5. C. D. HARMON II, R. D. BUSCH, J.F. BRIESMEISTER, and R. A. FORSTER, "Criticality Calculations with MCNP™: A Primer," LA-12827-M, Los Alamos National Laboratory (1994).

6. O. W. HERMANN and R. M. WESTFALL, "ORIGEN-S: Scale System Module to Calculate Fuel Depletion, Actinide Transmutation, Fission Product Buildup and Decay, and Associated Radiation Source Terms," Oak Ridge National Laboratory, NUREG/CR-0200, Rev. 5, Vol. 2, Sect. F7 (September 1995).

7. M. BENEDICT, T. H. PIGFORD, and H. W. LEVI, *Nuclear Chemical Engineering*, 2nd ed., pp. 76-78, 135-142, McGraw-Hill Inc., New York (1981).

8. C.F. GERALD, and P. O. WHEATLEY, *Applied Numerical Analysis*, 5th ed., Addison-Wesley Publishing Co., Reading, Massachusetts (1994).

9. R. L. MOORE, B. G. SCHNITZLER, C. A. WEMPLE, R. S. BABCOCK, and D. E. WESSOL, "MOCUP: MCNP/ORIGEN Coupling Utility Programs," Idaho National Engineering and Environmental Laboratory, INEL-95/0523 RSICC Code PSR-365.

10. O. W. HERMANN, "COUPLE: Scale System Module to Process Problem-Dependent Cross Sections and Neutron Spectral Data for ORIGEN-S Analyses," Oak Ridge National Laboratory, NUREG/CR-0200, Rev. 5, Vol. 2, Sect. F6 (September 1995).

11. D. BOWEN and R. D. BUSCH, "Using ORIGEN and MCNP to Calculate Reactor Criticals and Burnup Effects," *Trans. Am. Nucl. Soc.*, 77, 223 (1997).

12. S. L. EATON, C. A. BEARD, K. B. RAMSEY, J. J. BUKSA, and K. CHIDESTER, "Development of Nonfertile and Evolutionary Mixed Oxide Nuclear Fuels for Use in Existing Water Reactors," LA-UR-97-1359, Los Alamos National Laboratory (April 1997).

13. R. D. BUSCH, "A Primer for Criticality Calculations with DANTSYS," LA-13265, Los Alamos National Laboratory (1997).

14. R. E. MACFARLANE and D. W. MUIR, "The NJOY Nuclear Data Processing System, Version 91," LA-12740-M, Los Alamos National Laboratory (Oct 1994).

15. D. M. ETTER, *FORTRAN77 With Numerical Methods for Engineers and Scientists*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, California (1992).

16. J. R. LAMARSH, *Introduction to Nuclear Engineering*, 2nd ed., pp. 65, 77, Addison-Wesley Publishing Co., Reading, Massachusetts (1983).

17. W. B. WILSON, T. R. ENGLAND, D. C. GEORGE, D. W. MUIR, and P. G. YOUNG, "Recent Development of the CINDER'90 Transmutation Code and Data Library for Actinide Transmutation Studies," *Proc. GLOBAL'95 Int. Conf. on Evaluation of Emerging Nuclear Fuel Cycle Systems*, Versailles, France, September 11-14, 1995, p. 848 (1995).

18. D. I. POSTON, and H. R. TRELLUE, "User's Manual, Version 1.00, for *Monteburns*, Version 3.01," LA-UR-98-2718, Los Alamos National Laboratory (June 1998).

19. O. W. HERMANN, S. M. BOWMAN, M. C. BRADY, and C. V. PARKS, "Validation of the Scale System for PWR Spent Fuel Isotopic Composition Analyses," ORNL/TM-12667, Oak Ridge National Laboratory (March 1995).

20. M. D. DEHART, M. C. BRADY, and C. V. PARKS, "OECD/NEA Burnup Credit Calculational Criticality Benchmark Phase I-B Results," NEA/NSC/DOC(96)-06 and ORNL-6901, Oak Ridge National Laboratory (June 1996).

21. *Chart of the Nuclides*, 15th ed., General Electric Company, San Jose, California (1996).

22. J. E. TURNER, *Atoms, Radiation, and Radiation Protection*, pp. 60, 64, McGraw-Hill, Inc., New York (1992).

23. H. GRUPPELAAR, H. TH. KLIPPEL, J. L. KLOOSTERMAN, J. E. HOOGENBOOM, P. F. A. DE LEEGE, F. C. M. VERHAGEN, and J. C. BRUGGINK, "Evaluation of PWR and BWR Assembly Benchmark Calculations," ECN-C--93-088, Netherlands Energy Research Foundation and Energieonderzoek Centrum Nederland (November 1993).

24. B. D. MURPHY, "Characteristics of Spent Fuel from Plutonium Disposition Reactors, Vol. 1: The Combustion Engineering System 80+ Pressurized-Water-Reactor Design," ORNL/TM-13170/V1, Oak Ridge National Laboratory (June 1996).

25. H. M. FISHER, "A Nuclear Cross Section Data Handbook," LA-11711-M, Los Alamos National Laboratory (December 1989).

26. J. D. COURT, J.S. HENDRICKS, and S.C. FRANKLE, "MCNP™ ENDF/B-VI Validation: Infinite Media Comparisons of ENDF/B-VI and ENDF/B-V," LA-12887, Los Alamos National Laboratory (December 1994).

27. *Management and Disposition of Excess Weapons Plutonium*, Committee on International Security and Arms Control, p. 1, National Academy of Sciences, National Academy Press, Washington D.C. (1994).

28. B. D. MURPHY, "Characteristics of Spent Fuel from Plutonium Disposition Reactors, Vol. 3: A Westinghouse Pressurized-Water Reactor Design," ORNL/TM-13170/V3, Oak Ridge National Laboratory (July 1997).

29. M. M. EL-WAKIL, *Powerplant Technology*, p. 448, McGraw-Hill, Inc., New York (1984).