1-1-2005

Transient behavior of a nuclear reactor coupled to an accelerator

Suresh Babu Sadineni
University of Nevada, Las Vegas

TRANSIENT BEHAVIOR OF A NUCLEAR REACTOR COUPLED TO AN ACCELERATOR

by

Suresh Babu Sadineni

Bachelor of Engineering
Andhra University College of Engineering, Visakhapatnam, INDIA
2000

Master of Science
University of Nevada, Las Vegas
2002

Doctor of Philosophy in Engineering
Department of Mechanical Engineering
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
December 2005

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

UNLV
UNIVERSITY OF NEVADA LAS VEGAS

Dissertation Approval
The Graduate College
University of Nevada, Las Vegas

October 27, 2005

The Dissertation prepared by

Suresh Babu Sadineni

Entitled

Transient Behavior of a Nuclear Reactor Coupled to an Accelerator

is approved in partial fulfillment of the requirements for the degree of

Ph.D. in Mechanical Engineering

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative

ABSTRACT

Transient Behavior of a Nuclear Reactor Coupled to an Accelerator

by

Suresh Babu Sadineni

Dr. William Culbreth, Examination Committee Chair
Associate Professor of Mechanical Engineering
University of Nevada, Las Vegas

Accelerator Driven Systems (ADS) present one of the most viable solutions for transmutation and effective utilization of nuclear fuel. Spent fuel from reactors will be partitioned to separate plutonium and other minor actinides to be transmuted in the ADS. Without the ADS, minor actinides must be stored at a geologic repository for long periods of time. One problem with ADS is understanding the control issues that arise when coupling an accelerator to a reactor. "ADSTRANS" was developed to predict the transient behavior of a nuclear reactor coupled to an accelerator. It was based on MCNPX, a radiation transport code developed at the LANL, and upon a numerical model of the neutron transport equation. MCNPX was used to generate the neutron "source" term that occurs when the accelerator is fired. ADSTRANS coupled MCNPX to a separate finite difference code that solved the transient neutron transport equation. A cylindrical axisymmetric reactor with steel shielding was considered for this analysis. Multiple neutron energy groups, neutron precursor groups and neutron poisons were considered. ENDF/B cross-section data obtained through MCNPX was also employed. The reactor was assumed to be isothermal and near zero power level.

Unique features of this code are: 1) it predicts the neutron behavior of an ADS for different reactor geometry, material concentration, both electron and proton particle accelerators, and target material, 2) it develops input files for MCNPX to simulate neutron production, runs MCNPX, and retrieves information from the MCNPX output files.

Neutron production predicted by MCNPX for a 20 MeV electron accelerator and lead target was compared with experimental data from the Idaho Accelerator Center and found to be in good agreement. The spatial neutron flux distribution and transient neutron flux in the reactor as predicted by the code were compared with analytical solutions and found to be in good agreement. Fuel burnup and poison buildup were also as expected. ADSTRANS is intended to be a valid tool for the simulation of neutron behavior in a nuclear reactor coupled to an accelerator for the transmutation of nuclear waste.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank several people who helped me in achieving my goal.

My sincere thanks goes to Dr. William Culbreth for educating me in nuclear technology and guiding me throughout this project. It was due to his encouragement that I was able to complete this dissertation. I would like to thank my committee consisting of Dr. Robert Boehm, Dr. Ajit Roy and Dr. Sahjendra Singh for their continued support and suggestions in this project.

A special thanks to Dr. Frank Harmon, Director, Idaho Accelerator Center, Pocatello, for his support in the experimental work and accepting to be on the committee as an adjunct faculty member. It was a great opportunity working at the Idaho Accelerator Center on benchmarking experiments.

My utmost gratitude goes to my grandparents (Sadineni Venkaiah and Ramatherdham), who always supported my zeal for higher education and encouraged me to complete the highest level of education. I strongly feel that without their encouragement and support, I would never have achieved my goal.

Thanks to my other group members, especially Robert and my friends, who helped directly or indirectly in completing this dissertation.

Finally, but most importantly, thanks to my wife Padmaja and other family members for their continued support and encouragement in completing this dissertation.

-Suresh B Sadineni

NOMENCLATURE

Symbol	Description
A_v	Avogadro's number
A_w	Atomic weight
ADS	Accelerator Driven Systems
B	Geometric buckling
C_i	Number density of i^{th} neutron precursor group
d	Extrapolated distance
D	Neutron diffusion coefficient
E	Energy of neutrons
eV	Electron volts
$\tilde{H}$	Extrapolated height
H	Height
I	Number density of iodine
$\vec{J}$	Neutron current density
k	Neutron multiplication factor
k_{eff}	Effective multiplication factor
l	Neutron lifetime in the reactor
L	Thermal diffusion length
n	Number density of neutrons
N	Atomic number density
P	Number density of promethium
$\tilde{r}_s, \tilde{R}$	Extrapolated radius

R	Radius
S	Accelerator neutron source term
S	Number density of samarium
t	Time
T	Reactor period
X	Number density of xenon

Greek Symbols

σ_e	Microscopic elastic cross-section
σ_i	Microscopic inelastic cross-section
σ_s	Microscopic scattering cross-section
σ_γ	Microscopic radiative capture cross-section
σ_f	Microscopic fission cross-section
σ_a	Microscopic absorption cross-section
σ_t	Microscopic total cross-section
ρ	Mass density
μ	Average value of the cosine of the angle at which neutrons are scattered
λ_{tr}	Transport mean free path
Σ_{tr}	Macroscopic neutron transport cross-section
Σ_s	Macroscopic neutron scattering cross -section
Σ_a	Macroscopic neutron absorption cross –section
Σ_f	Macroscopic neutron fission cross –section
Σ_t	Macroscopic neutron total cross –section
υ	Neutron speed
$\hat{\Omega}$	Neutron direction
$\vec{\nabla}$	Del operator
ν	Number of neutrons produced per fission

ϕ	Neutron flux
φ	Angular neutron flux
θ	Angle
χ_g	Fraction of fission neutrons in the g^{th} energy group
λ_i	Decay constant of i^{th} neutron precursor group
β_i	Fraction fission neutrons appearing from i^{th} neutron precursor group
γ_I	Fission yield of iodine
λ_I	Decay constant of iodine
γ_X	Fission yield of xenon
λ_X	Decay constant of xenon
γ_p	Fission yield of promethium
λ_P	Decay constant of promethium

CHAPTER 1

INTRODUCTION

The transmutation of long-lived radioactive nuclei of nuclear waste to stable or short lived species has been under consideration for many years. There has been a renewed interest since 1990 in accelerator-driven subcritical systems (ADS) for transmutation of nuclear waste[1]. Spent fuel from reactors will be partitioned to separate plutonium and other minor actinides and transmuted in the ADS. This material would otherwise need to be stored and contained at a geological repository sites for time periods exceeding 10,000 years.

In an ADS, neutrons produced by spallation reactions are multiplied through fission reactions in a subcritical blanket[1-4]. By inducing fission in the minor actinides separated from spent fuel, fission products with short half lives are generated and the byproducts require storage for a much shorter period of time than the original nuclear waste before decaying to safe levels. Neutron multiplication through fission is largely dependent upon k_{eff}, the neutron multiplication factor of the blanket material. An ADS is usually designed such that the core is subcritical, in other words, $0.95 < k_{eff} < 0.98$, to improve safety[6]. The commissioning of a future industrial ADS qualified to transmute large amounts of minor actinides and long-lived fission products will require numerous technological innovations sustained by extensive basic research and development in the

field of accelerators, spallation targets, fuels, and subcritical systems[7]. One of the problems in the design of an ADS is understanding the control issues that arise in coupling an accelerator to a sub-critical reactor. An effort is made in the current work to develop a simulation code which can predict the transient behavior of an ADS. In this chapter the importance of nuclear energy, nuclear waste transmutation and Accelerator Driven Systems (ADS) are explained.

1.1 Nuclear Energy

The increasing need for the energy worldwide has prompted scientists to search for an alternative source of energy for both electrical power generation and for transportation. Nuclear energy is one of the promising solutions for the world's energy problems due to the availability of almost unlimited nuclear fuel. The further expansion of nuclear power will be heavily based on the development of effective treatment of spent nuclear waste.

Nuclear fission and fusion are two important technologies that can generate useful power. Energy released by the splitting of the nucleus of certain heavy fissionable atoms (^{235}U, ^{239}Pu) is called "fission", and the energy released by the fusing of two nuclei of small atoms (^{2}H, ^{3}H) is called "fusion". The existing technology for nuclear power production in large scale is through fission. In a fission reaction, a neutron is absorbed by the nucleus of a heavy fissile atom resulting in the release of two fission fragments, approximately 2.5 neutrons, 200 MeV of energy, gamma rays, and other subatomic particles. These neutrons further interact with the other fissile nuclei and the fission reaction is continued, resulting in a "chain reaction". Nuclear reactors are designed to sustain controlled chain reactions in fissile uranium or plutonium and may convert the

kinetic energy released during the fission process into heat to generate electricity or hydrogen fuel.

1.2 Nuclear Waste

Like any other power generation method, nuclear energy also leads to the creation of some waste product, one that is highly radioactive and produces heat through radioactive decay. Nuclear waste is produced by various industries ranging from power production to the use of radiopharmaceuticals. The largest inventory of radioactive nuclear waste is generated by commercial reactors designed for the production of electricity. Spent fuel from reactors contains fission products and minor actinides which have half lives ranging from seconds up to billions of years. Only 1% - 2% of the total mass of the spent fuel consists of long-lived materials of environmental and proliferation-risk concern[8]. The conventional method of long-term storage of nuclear waste is in geological repositories. As an alternative to long-term storage, the long-lived isotopes in nuclear waste can be transmuted to stable or short lived isotopes by exposing them to a large neutron flux, as produced in an ADS. The transmutation of long-lived isotopes will greatly reduce nuclear waste management costs by considerably reducing the amount of material that must be safely stored for a long period of time. As shown in figure 1.1, irradiation of long-lived technetium-99 (half-life of 212,000 years) by neutrons will result in technetium-100. Technetium-100 undergoes complete radioactive decay into stable ruthenium within minutes. Similarly radioactive isotopes like ^{237}Np and ^{238}Pu can be transmuted in the presence of neutron environment as shown. Since the long-lived radioactive isotopes account for only a small percent of spent nuclear fuel, spent fuel

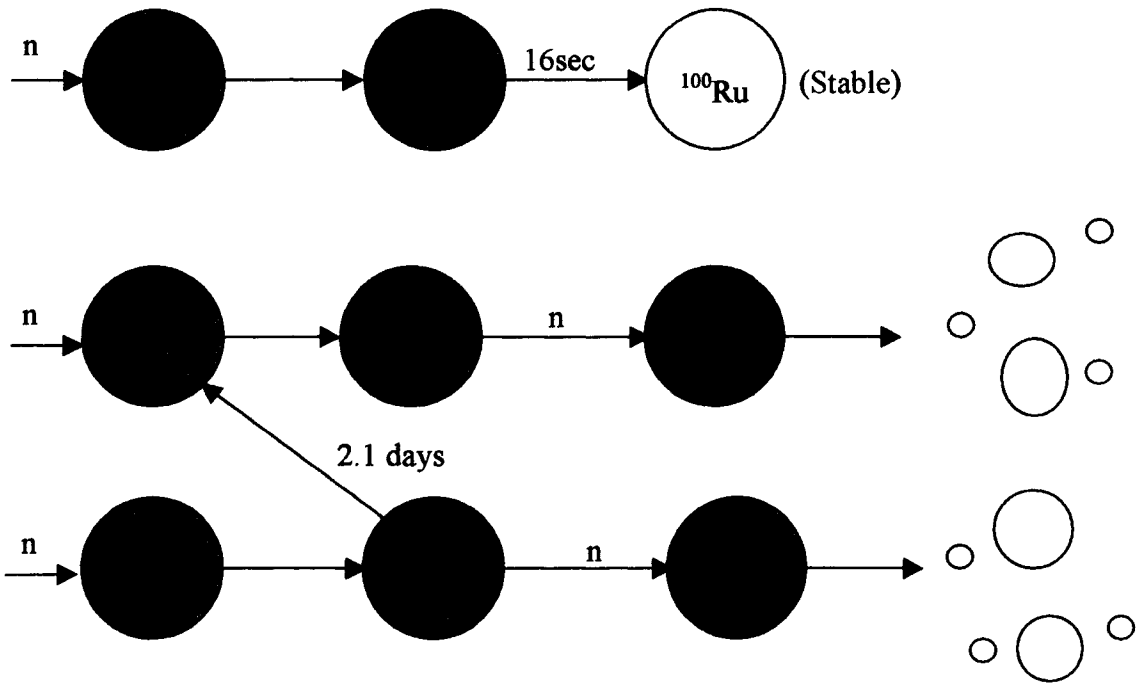


Figure 1.1 Transmutation of Radioactive Isotopes in the Presence of Neutrons

should be reprocessed and the long-lived radioisotopes partitioned out before exposure to the neutron flux in an ADS. Transmutation can take place by the following two methods,

1. Place the partitioned or separated long-lived radioisotopes inside a critical fission reactor.

2. "Burn" the partitioned or separated long-lived radioisotopes in a subcritical reactor (dedicated to transmutation), which is driven by an external source of neutrons produced by an accelerator. These systems are generally referred to as Accelerator Driven Systems (ADS).

Even though transmutation of nuclear waste is possible in critical reactors, its effectiveness in transmutation is limited by the criticality requirements[20]. A substantial

amount of excess reactivity is required initially and this should be compensated for by suitable controls (burnable poisons, control rods). As an alternative, recent technological developments in the field of accelerators have made it possible to generate high energy particle beams which can efficiently produce high energy neutrons required for transmutation. By using the ability of some of the minor actinides to undergo fission, the neutron population generated by the accelerator can be multiplied. The resulting subcritical reactor forms the basis of an accelerator-driven system (ADS), a good choice for the transmutation of nuclear waste.

1.3 Accelerator Driven Systems (ADS)

There are three fundamental components in an ADS: 1) a particle accelerator (proton, or electron), from which a high energy particle beam is produced; 2) a target (e.g. lead, bismuth, tungsten), where neutrons are produced from the particle beam; and, 3) a subcritical reactor, where neutron multiplication and transmutation takes place. The schematic of an ADS is shown in the figure 1.2.

1.3.1 Particle Accelerator

A particle accelerator is a device, which can accelerate charged particles from low energy to high energy, as high as TeV (10^{12} eV). There are different types of accelerators based on the type of particle they are accelerating (electron, proton, etc.), and based on different working principles (linear, cyclic etc.). Proton and electron accelerators have been proposed for ADS demonstration projects. Proton accelerators are more efficient in neutron production when compared to electron accelerators. For the same particle

energy, a proton accelerator can produce a few orders of magnitude higher number of neutrons than an electron accelerator. The advantage of an electron accelerator is that it

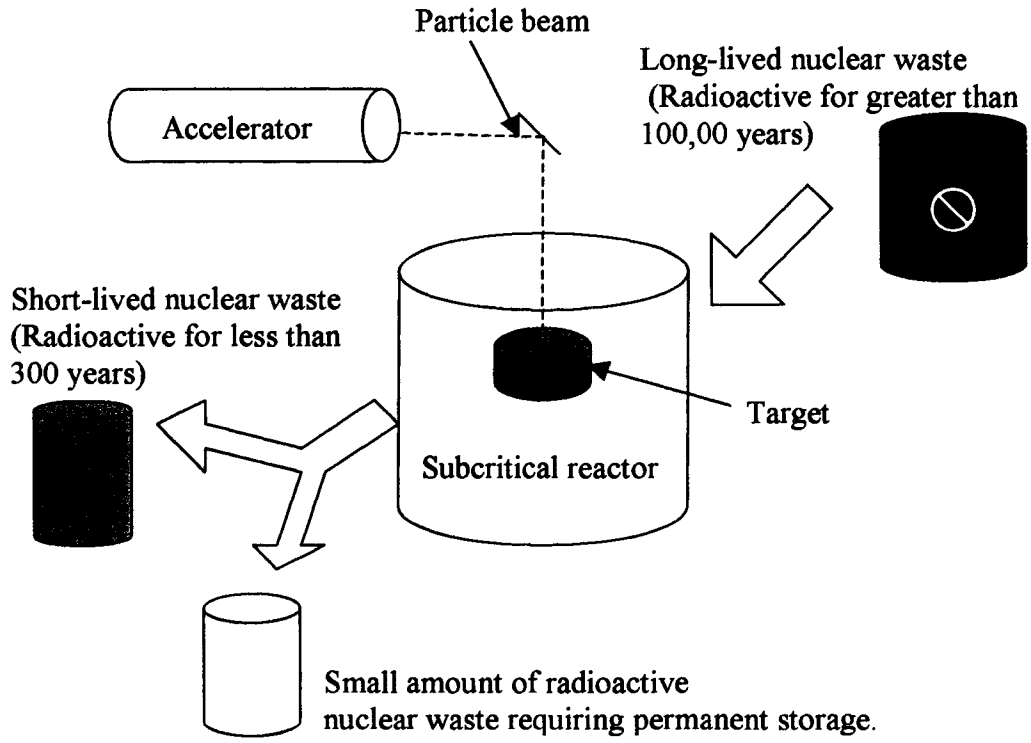


Figure 1.2 Diagram of an Accelerator Driven System (ADS).

is more compact than a proton accelerator. The neutron production from a proton accelerator is through a spallation reaction (p,n), where a proton is absorbed by a target nucleus resulting in the release of one or more neutrons. The neutron production from an electron accelerator is through (γ,n) reaction, in which the incoming electrons produce bremsstrahlung photons (γ) which in turn interact with the nucleus of an atom to cause it to decay through the emission of a neutron.

The Monte Carlo N-Particle radiation transport code, MCNPX, developed by Los Alamos National Laboratory, was used to predict the neutron production from a target when a high energy beam from an accelerator impinged upon the target. To complement this work, experiments were performed at the Idaho Accelerator Center, Pocatello in the year 2002, under the guidance of Dr. Frank Harmon. Neutron production was monitored in a lead target coupled to an 18 MeV electron accelerator. The neutron production rates were compared with MCNPX simulations and found to be in good agreement. This verified the nuclear cross section data used by MCNPX for neutron production from high-energy electrons.

1.3.2 Target

The target is one of the important components of an ADS. It is located inside the subcritical reactor, where the high energy particle beam impinges and produces neutrons. The target material should be selected based on the maximum neutron yield from the available particle beam energy. As shown in the figure 1.3, the general trend is a decrease of neutron threshold energy at the higher atomic numbers (Z). Hence, neutron yield will be higher from high Z materials like lead and bismuth. This has been verified through MCNPX simulations and through experimental work at the Los Alamos National Laboratory.

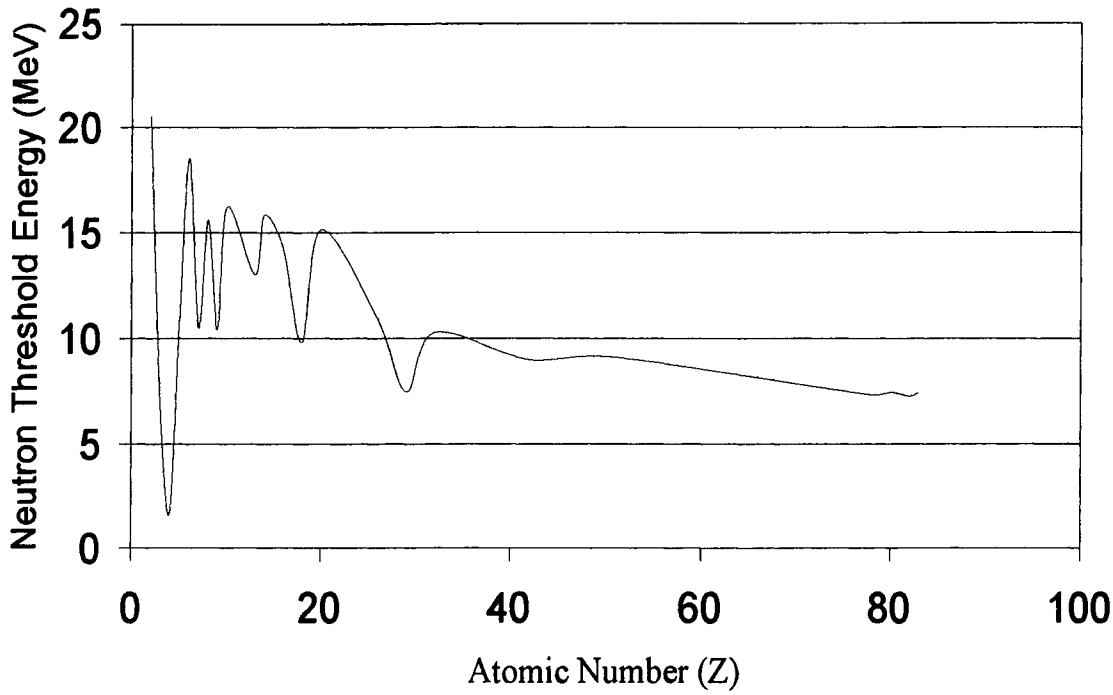


Figure 1.3 Dependence of Neutron Threshold Energy on Atomic Number.

1.3.3 Subcritical Reactor

The subcritical reactor core in an ADS is made out of partitioned radioisotopes from spent fuel. This includes the minor actinides, such as Np, Pu, Cm, and Am. The effective multiplication factor (k_{eff}) of a reactor is defined as the ratio of neutrons produced in one generation to the previous generation. If the value of the k_{eff} for a reactor is less than one, the reactor is subcritical, greater than one it is supercritical and equal to one it is a critical reactor with a steady-state neutron population. An ADS is usually designed such that the core is subcritical, in other words, $0.95 < k_{eff} < 0.98$, to improve safety. In a subcritical ADS, the fission process cannot be sustained without an external source of neutrons provided by the accelerator and its target. As opposed to a typical nuclear reactor, the fission rate in an ADS cannot continue to rise above steady-state

without the accelerator. This increases the ability of an ADS to safely consume plutonium, curium, and other fissionable material.

In this work, an effort was made to understand the control issues of an ADS by developing a simulation code "ADSTRANS" to model its performance. ADSTRANS simulates the transient behavior of an ADS for different dimensions, material concentrations, different types of particle accelerators and accelerator parameters. The code couples MCNPX to a separate finite difference code for transient neutron transport. The code can predict the neutron flux distribution in the reactor, transient flux in the reactor, fission product distribution, buildup of fission products before and after reactor shutdown, and provide other important information about the ADS reactor behavior. The code was developed to generate its own MCNPX input files for predicting k_{eff} and neutron production from target based on user parameters. ADSTRANS then runs the MCNPX simulation, reads the required information from the MCNPX output files, and uses this information along with the neutron transport equations to predict the transient behavior. The predictions from the code were in good agreement with analytical solutions and experimental results verified the use of MCNPX to predict neutron production from accelerators. This work is unique in a sense that there are no other codes developed to predict the transient behavior of a reactor coupled to an accelerator.

CHAPTER 2

LITERATURE REVIEW

In this chapter, a brief review is presented of the literature on transmutation of nuclear waste and the use of accelerator-driven subcritical systems. Current international projects that are underway in accelerator driven systems are also presented.

Nuclear reactors worldwide are producing nuclear waste, which is highly radioactive with radioisotope half-lives that extend up to billions of years. Current plans for geologic storage of waste require management of these wastes for long periods of time. In the United States, about 100 light water reactors (LWR), which produce approximately 20% of the nation's electricity, will create 87,000 tons of nuclear waste over the course of their lifetimes. Sixty thousand tons of spent fuel along with another 10,000 tons of defense waste is destined for geological disposal at the Yucca Mountain site in Nevada. Ironically, only about 1% - 2% of the total mass of spent nuclear fuel contains the long-lived radioisotopes of environmental and proliferation-risk concern.[8] Hence, these long-lived materials can be partitioned from the spent fuel and can be transmuted to short-lived or stable isotopes. Accelerator-driven nuclear reactors are one of the most viable solutions for transmutation. Although accelerator-driven transmutation has been proposed before under different circumstances [56,57,58], it has been feasible only due to the advent of high power, high current accelerators. There has been a renewed interest in the

transmutation of nuclear waste in accelerator-driven nuclear reactors since the early 1990's due to public opposition of the direct disposal of radiotoxic waste in geologic repositories. In 1991, G. P. Lawrence considered proton linear accelerators driving high-flux spallation neutron sources for nuclear waste transmutation and for the production of tritium. A proton beam energy of 1.6 GeV and current of 250 mA incident on a lead or lead-bismuth (Pb-Bi) spallation target generated a high flux of thermal neutrons which could be used in a surrounding blanket to produce tritium through the $^{6}Li(n,\alpha)T$ or $^{3}He(n,p)T$ reactions. The same process could be used to burn long-lived actinides and fission products in nuclear waste through radiative capture and the fission process.[14] In 2002, M. Lowenthal et al., made a simple analysis to conclude that transmutation can reduce the quantity of actinide waste. This is important since actinides are less attractive as weapons materials after recycling in reactors. Transmutation could reduce these radiological and proliferation hazards.[22]

Recent technological developments in the field of accelerators made it possible to generate high energy particle beams which can efficiently produce high energy neutrons, the basic components of the transmutation. In 1992, C. D. Bowman et al., described a new approach for commercial nuclear energy production without a long-term high-level waste stream and for the transmutation of both fission products and higher actinides in commercial waste. His process would use a flux of accelerator-produced neutrons in the 10^{16} n/cm^2-s range. Continuous neutron fluxes at this intensity, which is approximately 100 times larger than typical levels in a commercial thermal reactor, are possible due to recent advances in proton linear accelerator technology and to enhanced spallation target-moderator design.[10] In 1998, C. D. Bowman published a study that reviewed the

performance of a more highly developed reactor designed based on thermal and fast neutron spectra.[11] S. M. Seltzer, in 1973 made some calculations on the yield of photoneutrons from thick targets bombarded with electron beams. Yields were calculated from incident electron energies from 20 MeV down to the photonuclear cross section threshold for tantalum and tungsten targets.[12] William P. Swanson, in 1978, also made some calculations for low-energy neutrons released by electrons incident on semi-infinite targets of natural C, Al, Fe, Ni, Cu, Ag, Ba, Ta, W, Au, Pb, and U.[13] G. J. Van Tuyle, in 1998 made some interesting calculations about the neutron populations in subcritical reactors based on the particle beam parameters and level of subcriticality. He calculated that for a 100 mA driven system with an neutron-to-proton production ratio of 35 and k_{eff} of 0.95, ~ 4.4 x 10^{20}, neutrons would be generated every second, with ~95% of those produced from fission. He also calculated the thermal energy to be 53,000 MW by assuming 2.5 neutrons/fission, at 200 MeV/fission. His remark based on these observations was that one could drive a subcritical target (k_{eff} = 0.95) at power levels exceeding the largest nuclear power plants in operation.[5]

Transmutation will reduce the long-term radiotoxicity of the waste along with thermal loading in geological repositories. Transmutation can take place in critical fission reactors or in accelerator-driven subcritical systems. Transmutation in critical fission reactors is limited by the amount of excess reactivity that has to be supplied initially and compensated for by control poisons. F. Venneri et al., in 2000 proposed the use of accelerator driven systems for the transmutation of nuclear waste. They opinioned that critical fission reactors can transmute, but are limited by the excessive criticality

requirements. They also explained the advantages of using transmutation before sending nuclear waste to geological repositories.[20]

Even though transmutation is considered as an alternative for direct disposal of nuclear waste, there has not been enough experimental research done in this area. The key issue in the accelerator-based transmutation is the coupling of an accelerator to a subcritical reactor composed of fission products and minor actinides separated from spent nuclear fuel. The most important aspect in coupling an accelerator to a subcritical reactor is understanding the dynamics of the subcritical reactor. To complement these strategies efforts are underway for experimental demonstrations. Europe's MUSE demonstration project will couple a proton accelerator to a subcritical blanket.[7] The U.S. Reactor-Accelerator Coupling Experiments (RACE), as part of Advanced Fuel Cycle Initiative (AFCI) project, will couple an electron accelerator to a subcritical reactor to understand the transient behavior of the reactor[52,53,54]. Research and analysis is underway to study accelerator-driven systems before they can be experimentally tested. J. U. Knebel et al., in 1999 critically evaluated the characteristics of an accelerator-driven subcritical reactor system (ADS) and it's potential to transmute minor actinides and long-lived fission products safety.[15] Knebel also worked on core design, neutronics, safety, system analyses, materials and corrosion for ADS in 2000.[16] In 2003, Y. Kim et al., introduced and characterized an importance property of the external spallation neutrons in an accelerator-driven system (ADS) to address the source multiplication in a subcritical blanket. They evaluated the source importance function with a neutron transport code system.[6] K. Nishihara et al., in 2002 proposed a blanket design for the transmutation of ^{129}I along with the other minor actinides in accelerator-driven systems. They made some

interesting observations on the level of transmutation per year (250 kg/yr of minor actinides and 56 kg/yr of iodine) using ADS. They also discussed the benefits of the transmutation of iodine coupled with the underground disposal concepts.[35] S. Dulla et al., in 2004 determined the importance of transport effects in subcritical systems driven by an oscillated neutron source. They solved the transport equation using the discrete ordinates method.[9]

To help analyze nuclear reactors, several analytical and numerical models have been developed. The accuracy of these predictions depends upon the accuracy of the nuclear data that the calculations are based upon. G. Aliberti et al., in 2004 analyzed the impact of nuclear data uncertainty on performance parameters of reactor cores dedicated to the transmutation of radioactive wastes. They also provided guidelines on properties for new evaluation or validation experiments and cited required accuracies on specific nuclear data.[18] A. V. Voronkov et al., in 2004 developed a second order, semi-implicit numerical method for solving the multigroup nonstationary transport equation. The corresponding code was developed in two-dimensional R-Z geometry. They also compared their results with the analytical test problems.[19]

CHAPTER 3

THEORY

Energy released in a fission reaction serves as the source of energy in a nuclear reactor. Neutrons are the drivers of these fission reactions, and hence, the study of neutron behavior in reactors is an important area of nuclear reactor study. In this chapter, basic concepts on neutron interaction with matter are reviewed. Concepts in the nuclear reactor theory, such as neutron diffusion, radiation transport, time dependent multi-energy group neutron transport equations, delayed neutron precursor groups, boundary conditions of the reactor, neutron poisons in the reactor and basic geometries of nuclear reactors are discussed.

3.1 Basic Concepts

Basic concepts and terminology used in the nuclear reactor theory, such as neutron interaction with the matter, neutron cross-sections, number density calculations and fission reaction rates, are discussed in this section. Throughout this discussion, neutron energy is expressed in units of electron volts or eV where 1 eV = 1.602×10^{-19} J. An electron volt is defined as the energy that one electron has when exposed to a potential difference of one volt.

3.1.1 Neutron Interactions

The dynamics of a transmuter are dependent upon the distribution and energy of neutrons within the reactor fuel. Neutrons are electrically neutral and can penetrate deep into matter compared to charged particles, such as electron and protons. Neutrons interact with the nucleus of an atom through scattering or absorption interactions, with varying probabilities of occurrence. Important modes of interaction include elastic scattering, inelastic scattering, radiative capture, fission, and neutron absorption.

In elastic scattering, the collision of a neutron with a nucleus results in the emission of a single neutron while energy and momentum are conserved. The incident particle and the emitted particle are often indicated using the shorthand nomenclature, [n,n]. In an inelastic scattering collision, some of the original neutron energy is lost through the production of a gamma ray. Neutron absorption interactions are again divided into radiative capture and fission. In radiative capture, [n,γ], a neutron is absorbed by the nucleus which releases one or more γ-rays. In a fission reaction, [n,f], a neutron collides with the unstable nucleus of certain heavy atom, to split it apart into two or more fission fragments with the emission of neutrons, gamma rays, and other subatomic particles.

3.1.2 Cross-Sections

The probability of an interaction occurring between a neutron and a nucleus are quantified in terms of the effective target area of the nucleus expressed as a cross-sections. Microscopic cross-sections are denoted by the symbol σ, and are expressed in units of barns, abbreviated as b, where 1 barn is equal to 10^{-24} cm^2 of effective target area per nucleus. Different interaction cross-sections are denoted as σ_e for elastic scattering

cross-section, σ_i for inelastic scattering, σ_γ for radiative capture, or σ_f for fission cross-section. The sum of all these cross-sections is called as total cross-section, σ_t and is given as

$$\sigma_t = \sigma_e + \sigma_i + \sigma_\gamma + \sigma_f ..... \tag{3.1}$$

The sum of elastic and inelastic scattering cross-sections is called the scattering cross-section σ_s and is given as

$$\sigma_s = \sigma_e + \sigma_i \tag{3.2}$$

The sum of radiative capture and fission cross-sections become the absorption cross-section, σ_a and is given as

$$\sigma_a = \sigma_\gamma + \sigma_f ..... \tag{3.3}$$

Now the total cross-section σ_t can be expressed as the sum of scattering and absorption cross-section and is given as

$$\sigma_t = \sigma_s + \sigma_a \tag{3.4}$$

To quantify the total effective target area for an interaction within a volume of material, the atomic number density, N, is multiplied by the microscopic cross-section that defines the target area per nucleus to produce the macroscopic cross-section denoted by Σ given as

$$\Sigma = N\sigma \tag{3.5}$$

Since N and σ have units of cm^{-3} and cm^2 respectively, Σ has the units of cm^{-1}. The atomic density, N is given as

$$N = \frac{\rho A_v}{A_w} \tag{3.6}$$

where ρ is mass density in kg/m^3, A_w is the atomic weight of the target material expressed in units of g/gmole. A_v is Avogadro's number of 6.023 x 10^{23} atoms/gmole.

3.1.3 Fission Reaction

Combining neutrons and protons together to create a nucleus results in the conversion of some of the mass of these subatomic particles into binding energy that holds the nucleus together. To liberate a nucleon (neutron or proton) from the nucleus through a collision with a subatomic particle or a gamma ray, this binding energy must be supplied by the incoming particle. The binding energy per nucleon is fairly low for heavy, fissionable nuclei like ^{235}U or ^{239}Pu. Neutrons with kinetic energy above the binding energy of nucleons within the nucleus can often cause the nucleus to split into smaller nuclei. In a typical fission reaction involving ^{235}U, approximately 2.42 neutrons, two large fission fragments, neutronios and subatomic particles, along with 200 MeV of energy are produced as shown in equation 3.7.

$$n + {}^{235}U \rightarrow Fission\ Fragments + 2.42\,neutrons + 200\,MeV \qquad (37)$$

Almost 85% of the energy released in a fission reaction appears as the kinetic energy of the fission fragments. Fission neutrons are produced with a wide range of energies. The average fission neutron energy is 1.98 MeV and most probable energy that any neutron has is 0.73 MeV. Fission neutrons produced in one generation will continue the fission reaction by interacting with the other fissionable nuclei. In a critical reactor, one neutron produced by fission survives in every generation to cause one additional nucleus to undergo fission. The 200 MeV energy produced in fission reactions is the source of energy from nuclear reactors. This energy is split between the kinetic energy of the

fission neutrons and fission products, gamma rays, and some is lost to the production of neutrinos which are unrecoverable.

The study of neutron behavior within a nuclear reactor is an important section of nuclear reactor analysis. Neutrons produced at one part of the reactor by fission may go on to cause subsequent fissions. Unfortunately, some will penetrate into the materials surrounding the reactor to be absorbed and lost from the neutron inventory. The transport of neutrons within a reactor is quantified through the neutron transport equation, which takes into account the production, loss, and diffusion of neutrons through the reactor core. Neutron diffusion obeys Fick's Law of Diffusion and is an important mechanism for the distribution of neutrons throughout a reactor.

3.2 Neutron Diffusion

To model ADS behavior, the time-dependent of diffusion of neutrons throughout the reactor core must be computed. The diffusion of neutrons is combined with predictions of loss by absorption and generation through fission to create a general equation for neutron transport. Unlike other diffusion processes (heat, gas molecules), simple diffusion theory has limited validity for neutrons. Diffusion theory is not valid under the following conditions:[36]

1. In a medium that strongly absorbs neutrons
2. Within about three mean free paths of either a neutron source or the surface of a medium
3. When the scattering of neutrons is strongly anisotropic.

Nevertheless, diffusion theory is often used to estimate the reactor properties. Neutron diffusion theory is based on Fick's Law of diffusion, which is generally used to account for chemical diffusion. Fick's Law for neutron diffusion, for example, in x-direction is given as

$$J_x = -D\frac{d\phi}{dx} \tag{3.8}$$

where J_x is the neutron current density in the x-direction and ϕ is the neutron flux. Both the neutron current and flux have units of neutrons/cm^2-sec, although neutron current density is a vector quantity and flux is a scalar. D is called the neutron diffusion coefficient which has the units of centimeters. Diffusion coefficient, D, is approximated as

$$D = \frac{\lambda_{tr}}{3} \tag{3.9}$$

where λ_{tr} is the transport mean free path and is given by:

$$\lambda_{tr} = \frac{1}{\Sigma_{tr}} = \frac{1}{\Sigma_s(1-\mu)} \tag{3.10}$$

Here, Σ_{tr} is the macroscopic neutron transport cross-section, Σ_s is the macroscopic neutron scattering cross-section, and μ is the average value of the cosine of the angle at which neutrons are scattered in the medium. The value of μ at most of the energies of interest in reactor calculations can be computed from the following simple formula

$$\mu = \frac{2}{3A} \tag{3.11}$$

where A is the atomic mass number. The distance traveled by neutrons before they are absorbed is called the thermal diffusion length, represented by L, and is given as

$$L = \sqrt{\frac{D}{\Sigma_a}} \tag{3.12}$$

where Σ_a is the macroscopic neutron absorption cross-section. It can be understood from the above equation that the greater the value of L, the further the neutrons travel before they are absorbed, which means the medium is more diffusive and less absorptive.

3.3 Neutron Transport/Diffusion Equation

The most general case of the neutron transport equation with position vector, **r**, energy, E, direction $\hat{\Omega}$ and time, t, dependence is[37]

$$\frac{\partial n}{\partial t} + \upsilon\hat{\Omega}\cdot\vec{\nabla}n + \upsilon\overline{\Sigma}_t n(r,E,\hat{\Omega},t) = \int_{4\pi} d\hat{\Omega}'\int_0^\infty dE'\upsilon'\Sigma_s(E'\to E,\hat{\Omega}'\to\hat{\Omega})n(r,E',\hat{\Omega}',t) + s(r,E,\hat{\Omega},t) \tag{3.13}$$

where n(**r**,E, $\hat{\Omega}$,t) is the neutron density with seven independent variables (**r** = **x, y, z;** E; $\hat{\Omega}$= θ, φ; t), υ is the speed of the neutrons, and Σ_t and Σ_s are the macroscopic total and scattering cross-sections, respectively. The first term in this equation represents the time-rate-of-change in neutron density within the control volume. The second term represents the net neutron leakage through diffusion, the third term represents the loss due to collisions, and the fourth term is the gain due to in-scattering. The last term, s(**r**,E, $\hat{\Omega}$,t), quantifies the rate that neutrons are produced by fission or decay. The neutron transport equation in terms of angular flux is

$$\frac{1}{\upsilon}\frac{\partial\varphi}{\partial t} + \hat{\Omega}'\cdot\vec{\nabla}\varphi + \Sigma_t(r,E)\varphi(r,E,\hat{\Omega},t) = \int_{4\pi} d\hat{\Omega}'\int_0^\infty dE'\Sigma_s(E'\to E,\hat{\Omega}'\to\hat{\Omega})\varphi(r,E',\hat{\Omega}',t) + s(r,E,\hat{\Omega},t) \tag{3.14}$$

where $\varphi(\mathbf{r},E,\Omega,t)$ is the angular neutron flux, while all other symbols are as described with the earlier equations.

For most of the reactor calculations, the angular dependence of the flux is not important and the neutron transport equation may be solved to find ϕ is function of spatial position **(r),** energy (E), and time (t). In this case, the transport equation further simplifies to the following form which is know as the neutron continuity equation.

$$\frac{1}{\upsilon}\frac{\partial\phi}{\partial t}+\vec{\nabla}\cdot\vec{J}(r,E,t)+\Sigma_t(r,E)\phi(r,E,t)=\int_0^\infty dE'\Sigma_s(E'\rightarrow E)\phi(r,E',t)+S(r,E,t) \quad (3.15)$$

where $\vec{J}(r,E,t),\phi(r,E,t)$ are the neutron flux and neutron current density, respectively.

3.3.1 Diffusion Approximation

As explained in the previous section, it is a common practice to assume a simplified neutron diffusion process in determining neutron transport within a reactor. The neutron current density, $\vec{J}$, can be expressed in terms of the neutron flux, ϕ, with the diffusion approximation represented by:

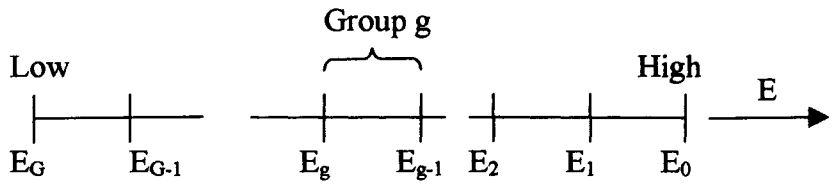
$$\vec{J}(r,E,t)=-D(r,E,t)\vec{\nabla}\phi(r,E,t) \quad (3.16)$$

In this expression, $D(r,E,t)$ is the neutron diffusion coefficient. By substituting $\vec{J}$ from equation 3.16 into equation 3.15, an equation with one dependent variable, ϕ, can be obtained. The resulting equation is called the energy-dependent neutron diffusion equation, and is given by:

$$\frac{1}{\upsilon}\frac{\partial\phi}{\partial t}-\vec{\nabla}\cdot D(r,E)\vec{\nabla}\phi+\Sigma_t(r,E)\phi(r,E,t)=\int_0^\infty dE'\Sigma_s(E'\rightarrow E)\phi(r,E',t)+S(r,E,t) \quad (3.17)$$

3.3.2 Multi-group Diffusion Equation

The neutron energies typically available in a reactor span the range of 10^{-3} to 10^{7} eV. The probability of nuclear reactions expressed in terms of cross-sections is dependent upon the neutron energy. However, it is not feasible to track every neutron through every collision and monitor the changes in its energy. To analyze the large number of neutrons that typically exist in a nuclear reactor, the neutrons are divided into groups based on their energy. Each group is represented by averaged cross-sections within the energy band of that group. In a one-speed (one group) approximation, all neutrons within the reactor are assumed to have the same cross-sections for fission, absorption, and scattering. This greatly simplifies the analysis, but lacks accuracy. The success of one-speed calculation depends on the choice of the one-speed cross-sections that are used in the neutron transport equation. Practical reactor models require a more realistic treatment of the neutron energy dependence and multiple energy groups are employed. Most reactor calculations achieve sufficient accuracy using only a few energy groups. The following is the schematic of "g" neutron energy groups, and the corresponding multigroup diffusion equation. As neutrons scatter off of nuclei within the reactor, they loose energy. The initial fission neutron energy, E_o shown in the schematic below is 2 MeV and multiple collisions will eventually decrease the energy to thermal energy levels at about 0.025 eV.



$$\frac{1}{\upsilon_g}\frac{\partial \phi_g}{\partial t} - \vec{\nabla}\cdot D_g \vec{\nabla}\phi_g + \Sigma_{tg}\phi_g(r,t) = \sum_{g'=1}^{G}\Sigma_{sg'g}\phi_{g'} + \chi_g \sum_{g'=1}^{G} v_{g'}\Sigma_{fg'}\phi_{g'} + S_g \qquad (3.18)$$
$$g = 1,2,\ldots\ldots\ldots,G$$

In this equation, υ_g, D_g, Σ_{tg}, $\Sigma_{sg'g}$, $\Sigma_{fg'}$, χ_g, and $v_{g'}$ are the group constants. υ_g is the neutron speed in the "g'th" energy group. D_g is the diffusion coefficient, Σ_{tg} is the total cross-section, of g^{th} energy group. χ_g is the fraction of fission neutrons that are produced in the "g" energy group. $\Sigma_{sg'g}$ is the group transfer cross-section. $\Sigma_{fg'}$ is the fission cross-section, and $v_{g'}$ is the number of neutrons produced. These terms may be found using equations 3.19 through 3.25:

$$\phi_g(r,t) \equiv \int_{E_g}^{E_{g-1}} dE\phi(r,E,t) \qquad (3.19)$$

$$\Sigma_{tg} \equiv \frac{1}{\phi_g}\int_{E_g}^{E_{g-1}} dE\Sigma_t(E)\phi(r,E,t) \qquad (3.20)$$

$$D_g \equiv \frac{\int_{E_g}^{E_{g-1}} dED(E)\nabla_j\phi(r,E,t)}{\int_{E_g}^{E_{g-1}} dE\nabla_j\phi(r,E,t)} \qquad (3.21)$$

$$\frac{1}{\upsilon_g} \equiv \frac{1}{\phi_g}\int_{E_g}^{E_{g-1}} dE\frac{1}{\upsilon}\phi(r,E,t) \qquad (3.22)$$

$$\int_{E_g}^{E_{g-1}} dES_f(r,E,t) = \int_{E_g}^{E_{g-1}} dE\chi(E)\left[\sum_{g''=1}^{G}\int_{E_{g'}}^{E_{g'-1}} dE'\nu(E')\Sigma_f(E')\phi(r,E',t)\right] \qquad (3.23)$$

$$v_{g'}\Sigma_{fg'} \equiv \frac{1}{\phi_{g'}}\int_{E_{g'}}^{E_{g'-1}} dE'\nu(E')\Sigma_f(E')\phi(r,E',t) \qquad (3.24)$$

$$\chi_g \equiv \int_{E_g}^{E_{g-1}} dE\chi(E) \qquad (3.25)$$

Group transfer cross sections:

The most general equation for calculating the group transfer cross-section is

$$\Sigma_{sg'g} \equiv \frac{1}{\phi_{g'}} \int_{E_g}^{E_{g-1}} dE \int_{E_{g'}}^{E_{g'-1}} dE' \Sigma_s(E' \to E)\phi(r, E', t) \tag{3.26}$$

which can be simplified largely by considering the most possible situations in the reactor. For neutrons of substantially greater energies than the thermal energy of the nuclei (typically less than 0.1 eV), the neutron could never gain energy in scattering collisions. This implies that fast neutrons will always slow down and only down scattering occurs. Hence, for fast groups (high energy), the group transfer cross-section is

$$\Sigma_{sg'g} = 0, \quad for\ g' > g \tag{3.27}$$

$$\sum_{g'=1}^{G} \Sigma_{sg'g}\phi_{g'} = \sum_{g'=1}^{g-1} \Sigma_{sg'g}\phi_{g'} + \Sigma_{sgg}\phi_g \tag{3.28}$$

This condition is particularly valid in the present work where only a few neutron energy groups are considered. We can further simplify the group transfer cross-section term by choosing the group spacing, such that, neutrons will scatter to the next lowest group, that is:

$$\sum_{g'=1}^{G} \Sigma_{sg'g}\phi_{g'} = \Sigma_{sg-1,g}\phi_{g-1} + \Sigma_{sgg}\phi_g \tag{3.29}$$

which makes the multigroup equations directly coupled. To achieve direct coupling one should consider the group spacing such that

$$\frac{E_{g-1}}{E_g} > \frac{1}{\alpha} \tag{3.30}$$

The rationale behind this is that a neutron of energy E cannot scatter below αE in a single scattering collision, where α is

$$\alpha = \left(\frac{A-1}{A+1}\right)^2, \tag{3.31}$$

and where A is the atomic mass number.

3.3.3 Boundary and Initial Conditions

Since solutions to the time-dependent neutron transport equation requires both initial and boundary conditions, suitable conditions must be specified. The initial condition is specified with a given distribution of neutrons in space and energy:

$$\phi(r,E,t) = \phi_0(r,E)$$

The boundary condition is based on "vacuum boundaries" where any neutron that escapes the reactor and its reflector may not reenter the reactor. Vacuum boundaries ($\vec{J}_- = 0$) were considered in the present work. The boundary condition for a vacuum boundary is represented by $\vec{J}_-(r_s,t) = 0$, $\vec{J}_-(z_s,t) = 0$, In a cylindrical geometry, this indicates that there is no incoming neutron current at the boundaries defined by r_s as the geometric radius and z_s as the top or bottom of the reactor. By solving for the corresponding neutron flux, ϕ, we can find $\phi(\tilde{r}_s) = \phi(\tilde{z}_s) = 0$. where $\tilde{r}_s$, $\tilde{z}_s$ are the extrapolated radius and height and are given as $\tilde{r}_s = r_s + d$, $\tilde{z}_s = z_s + d$. In this expression, d is the linear extrapolated distance which is given by $d = 2.13D$. D is the diffusion coefficient. In the present analysis, the linearly extrapolation distance, d, was used in both radial and axial directions.

3.4 Delayed Neutrons

The neutron population in a reactor changes exponentially with the reactor period T which is defined by the equation (without delayed neutrons) as

$$T = \frac{l}{k-1} \tag{3.32}$$

where l is the neutron lifetime and k is the neutron multiplication factor. k is the ratio of population of neutrons from one generation to the previous generation. If all fission neutrons are produced as prompt neutrons (during fission), then the neutron lifetime is very short (~ 0.1 milliseconds). A small increase in the neutron multiplication factor, k, will cause the reactor period, T, to change significantly. The neutron flux within a reactor changes as $\Phi/\Phi_o = e^{t/T}$, hence, the reactor will respond rapidly to changes in the multiplication factor which makes it difficult to control the reactor. Fortunately, a small fraction of the neutrons (~ 1%) that are produced from the decay of fission products appear with a time delay after the fission. These are referred to as delayed neutrons. Delayed neutrons play a very important role in the dynamics of a reactor.

Fission fragments are typically radioactive and produce neutrons through decay with varying half-lives. Since it is difficult to track all the fission isotopes, it has been customary to group these into six precursor groups based on their half-lives. The approximate half-lives of these precursor groups are 55, 22, 6, 2, 0.5 and 0.2 sec. The total delayed neutron yield and half-life data of each precursor group are available for different fissionable isotopes. The number densities of the precursor groups can be calculated from the following equation:

$$\frac{\partial C_i}{\partial t} = -\lambda_i C_i + \beta_i \nu \Sigma_f \phi \tag{3.33}$$

where C_i is the number density of the i[th] precursor group, λ_i is the decay constant, and β_i is the fraction of each group (both prompt and delayed) emitted per fission. By including the effect of delayed neutrons, the multi-group diffusion equation takes the form:

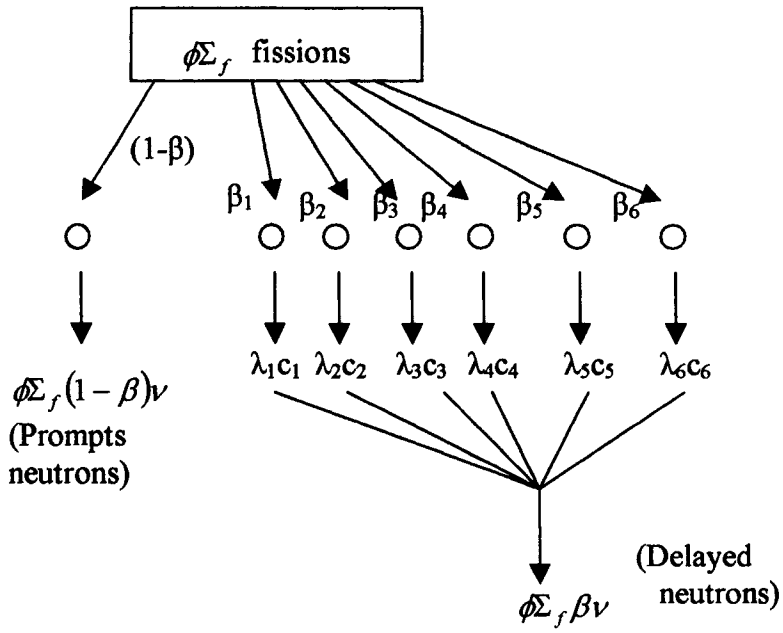


Figure 3.1 Prompt and Delayed Neutrons in a Transient Reactor.[47]

$$\frac{1}{\upsilon_g}\frac{\partial\phi_g}{\partial t} - \vec{\nabla}\cdot D_g\vec{\nabla}\phi + \Sigma_{tg}\phi_g(r,t) = \sum_{g'=1}^{G}\Sigma_{sg'g}\phi_{g'} + \chi_g\sum_{g'=1}^{G}(1-\beta)\nu_{g'}\Sigma_{fg'}\phi_{g'} + \sum_{i=1}^{6}\lambda_i C_i + S_g \quad g = 1.....G \tag{3.34}$$

3.5 Time Dependent Reactor Analysis

Even with the present day computers it is very difficult to analyze the reactor behavior over a wide range of time scales. Different numerical analyses require different time

steps to obtain dependable results. Time-dependent reactor analyses are divided into three different classes based on the required time step. These are short, intermediate, and long time analyses.

Short time analysis is usually done to understand reactor transient behavior and safety. This kind of analysis is usually done over a time interval of a few seconds to few minutes of reactor time. In this short period of time, fuel depletion is very small and may be neglected.

Intermediate time analysis is generally done to understand the production of fission products. This analysis is usually done over hours to few days of reactor time. Here, also, the depletion of fuel is very small and can be ignored.

Long time analysis is done to calculate fuel depletion and neutron flux variation and to determine the variation in power level inside the reactor. In the present work, a numerical model is employed with a time step of 0.1 millisecond to determine neutron flux, fuel depletion, and the buildup of poisons over time. This time step was decided based on the average prompt neutron lifetime in a typical reactor ($\sim 10^{-4}$ seconds).

3.6 Neutron Poisons in the Reactor

Each nuclear fission results in the production of two fission fragments, each containing about half of the total number of nucleons in the original nucleus. Fission products are typically radioactive and may also be bombarded by fission neutrons within the reactor. Some fission products have extremely high neutron absorption cross-sections and their production within the reactor can have a significant impact on reactor behavior by parasitically absorbing the available neutrons. As a result, the accumulation of these

fission products called "neutron poisons" can significantly change the multiplication factor and, hence, the neutron population in the reactor. Neutron poisons in the reactor build up over a period of time, and some isotopes continue to build up throughout reactor operation because of their small absorption cross-sections and long half lives. Some isotopes, like xenon (^{135}Xe) and samarium (^{149}Sm), have very large absorption cross-sections and relatively long half-lives. They reach an equilibrium concentration where their production and loss are equal. ^{135}Xe and ^{149}Sm have absorption cross-sections of $2.65 \cdot 10^6$ barns and 58,700 barns, respectively, for thermal neutrons (2,200 m/sec)[36]. Because of their very high absorption cross-sections ^{135}Xe, ^{149}Sm isotopes receive special attention in reactor transient analysis.

3.6.1 Xenon Concentration

^{135}Xe is formed from directly by fission and also as a result of the decay of ^{135}I. The ^{135}I isotope is formed by fission and by the decay of ^{135}Te. These processes and their half-lives are summarized below[36]. The fission yields of ^{135}I and ^{135}Xe for different fissile isotopes are given in table 3.1.

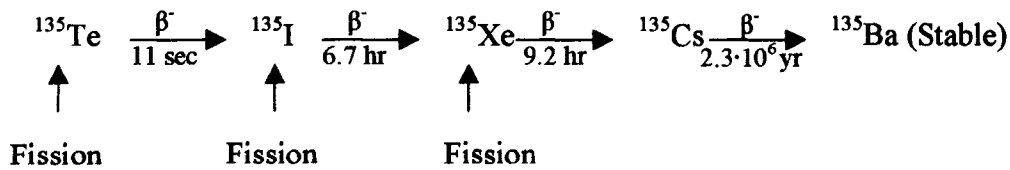


Table 3.1 Fission Product Yields (Atoms per Fission) from Thermal Fission[36]

Isotope	^{233}U	^{235}U	^{249}U
^{135}I	0.0475	0.0639	0.0604
^{135}Xe	0.0107	0.00237	0.0105
^{149}Pm	0.00795	0.01071	0.0121

Table 3.2 Decay Constants for Fission Product Poisons

Isotope	λ (1/sec)	λ (1/hour)
^{135}I	$2.87 \cdot 10^{-5}$	0.1035
^{135}Xe	$2.09 \cdot 10^{-5}$	0.0753
^{149}Pm	$3.63 \cdot 10^{-6}$	0.0131

Since ^{135}Te decays so rapidly to ^{135}I, it can be assumed to be produced directly by fission. ^{135}Xe is produced from the decay of ^{135}I, hence, the concentration of ^{135}Xe at any time depends upon the concentration of ^{135}I. The rate equations for the concentrations of ^{135}I and ^{135}Xe are given as follows:

$$\frac{dI}{dt} = \gamma_I \overline{\Sigma}_f \phi_T - \lambda_I I \tag{3.35}$$

$$\frac{dX}{dt} = \lambda_I I + \gamma_X \overline{\Sigma}_f \phi_T - \lambda_X X - \overline{\sigma}_{aX} \phi_T X \tag{3.36}$$

where I is the concentration of ^{135}I and X is the concentration of ^{135}Xe in atoms/cm^3. γ_I and γ_X are fission yields, λ_I, λ_X are decay constants of ^{135}I and ^{135}Xe, respectively, and Σ_a

is the thermal fission cross-section. ϕ_T is the thermal neutron flux and σ_{aX} is the thermal absorption cross section of ^{135}Xe.

If the reactor is shut down after operating for a long period of time at a constant flux, an examination of equations 3.35 and 3.36 reveals several consequences. First the removal of ^{135}Xe by neutron capture decreases to zero (since flux ϕ is zero). This leaves ^{135}Xe decay as the sole removal mechanism. However the production of ^{135}Xe from the decay of ^{135}I continues. Because of the short half-life of ^{135}I compared to ^{135}Xe, the ^{135}Xe concentration increases initially before it can decay out as shown in figure 3.2.

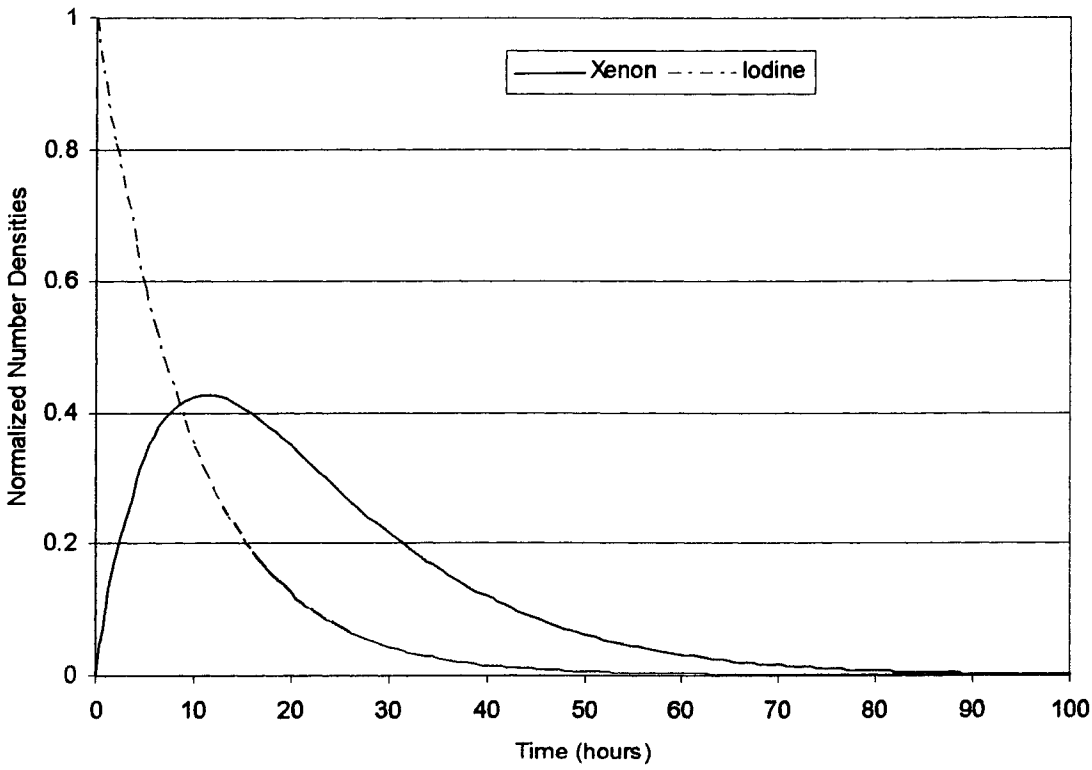
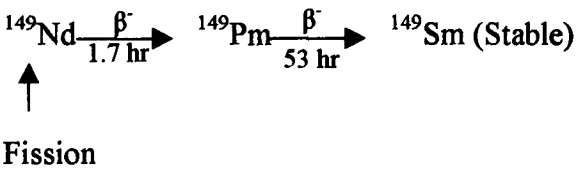


Figure 3.2 Behavior of ^{135}Xe and ^{135}I Following Reactor Shutdown at t=0

3.6.2 Samarium Concentration

The ^{149}Sm is not formed by fission, but from the decay of neodymium (^{149}Nd) produced by fission. The concentration of samarium is less concern with compared to xenon in the reactor calculations. The formation of ^{149}Sm isotopes from the decay chain is as follows[36]



Since the neodymium decays relatively quickly, it is assumed that promethium is produced directly from fission. The rate equations used to determine the concentrations of promethium and samarium are as follows[36]

$$\frac{dP}{dt} = \gamma_P \overline{\Sigma}_f \phi_T - \lambda_P P \tag{3.37}$$

$$\frac{dS}{dt} = \lambda_P P - \overline{\Sigma}_{aS} \phi_T S \tag{3.38}$$

where P is the concentration of ^{149}Pm, S is the concentration of ^{149}Sm in atoms/cm^3, γ_p is fission yield, λ_p is decay constant of ^{149}Pm and Σ_f is the thermal fission cross section, φ_T is the thermal neutron flux, and σ_{aS} is the thermal absorption cross section of ^{149}Sm.

The behavior of ^{149}Sm following reactor shutdown is governed by the decay of the accumulated ^{149}Pm. After shutdown, the removal by neutron capture is zero since flux ϕ is zero. Unlike ^{135}Xe, ^{149}Sm is a stable isotope, hence, it continues to build up after reactor shutdown until all promethium decays, and reaches a steady level.

3.7 Geometry of Reactors

One of the important factors that can affect the geometric design of a nuclear reactor is the neutron economy. Optimum geometry of a nuclear reactor should minimize neutron leakage out of the reactor. To minimize leakage, the ratio of surface area to reactor volume should be as small as possible. Two important ways to minimize neutron leakage are: 1) by providing neutron reflectors around the core, 2) by reducing the surface area of the core with respect to the volume. For the same mass, a spherical reactor will have least surface area compared to other geometries, including rectangular parallelepipeds and cylinder. However, due to practical considerations including manufacturability, spherical geometry is not suitable as a typical critical reactor. Cylindrical geometry

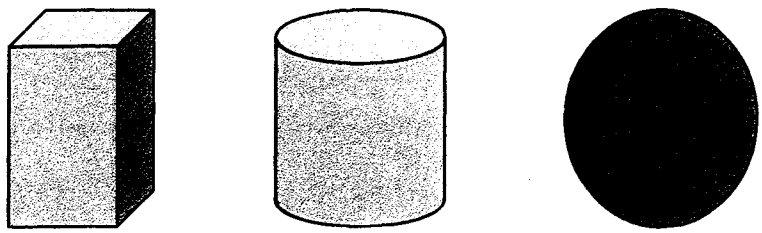


Figure 3.3 Basic Shapes of Nuclear Reactors

is the next optimum geometry after spherical geometry because of its minimal surface area. Due to this reason, most critical nuclear reactors are designed with a cylindrical geometry. Hence, cylindrical geometries are considered throughout this analysis.

CHAPTER 4

NUMERICAL SCHEME

In this chapter, geometric details of the reactor, details of the numerical scheme used, and numerical techniques used to solve the system of simultaneous equations are explained.

The time-dependant neutron diffusion equation with multi-energy groups was solved through an implicit finite difference technique. This required the solution of a set of simultaneous equations at every timestep.[46] In the present analysis, the time derivative was approximated by backward-differencing and the spatial derivatives were approximated by the central difference method. The cylindrical reactor was divided into finite volumes in both radial and axial directions. Due to symmetry, variations in the circumferential direction were not included in this axisymmetric analysis. The reactor was assumed to have homogeneous material distribution inside each finite volume. These assumptions greatly simplified the analysis and saved computational time. In the later part of this chapter, a detailed explanation is presented to compare the advantages of using a 2-D, axisymmetric coordinate system over a full 3-D Cartesian model.

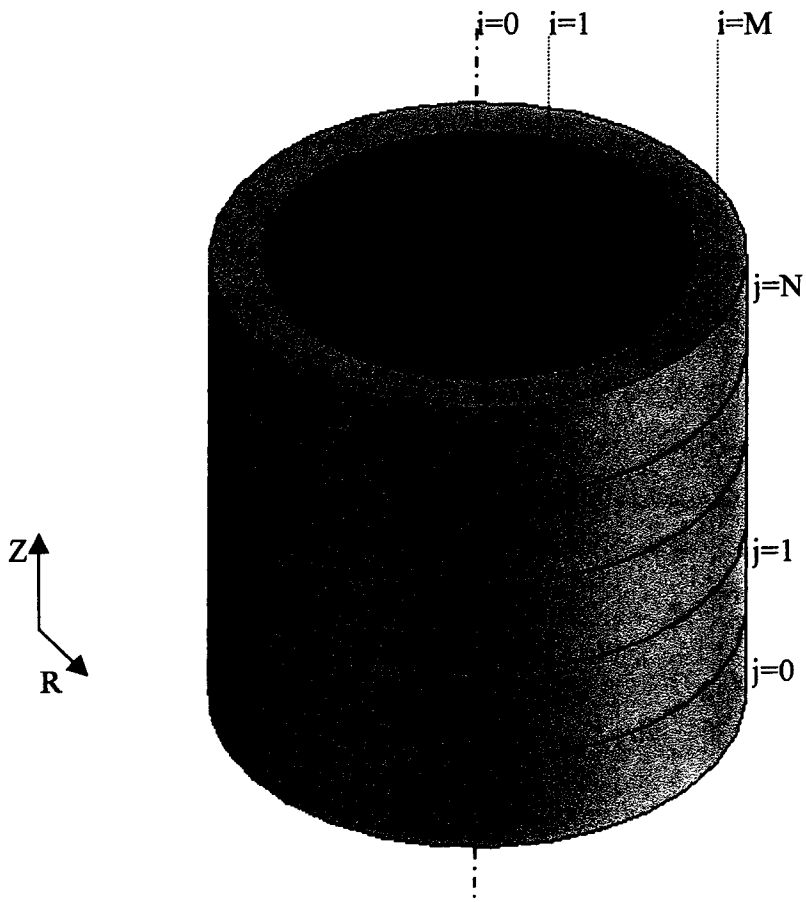


Figure 4.1 Cylindrical Reactor Discretized into Finite Volumes

4.1 Finite Volume Model

In this analysis, a cylindrical reactor core with its shielding was discretized into finite volumes in the radial and axial directions as shown in figure 4.1. A 2-D mesh system with the corresponding nodes are shown in figures 4.2, 4.3 and 4.4. Each volume within the mesh was in the form of a concentric ring. The center of each volume contains a node that was used to generate the finite difference equations.

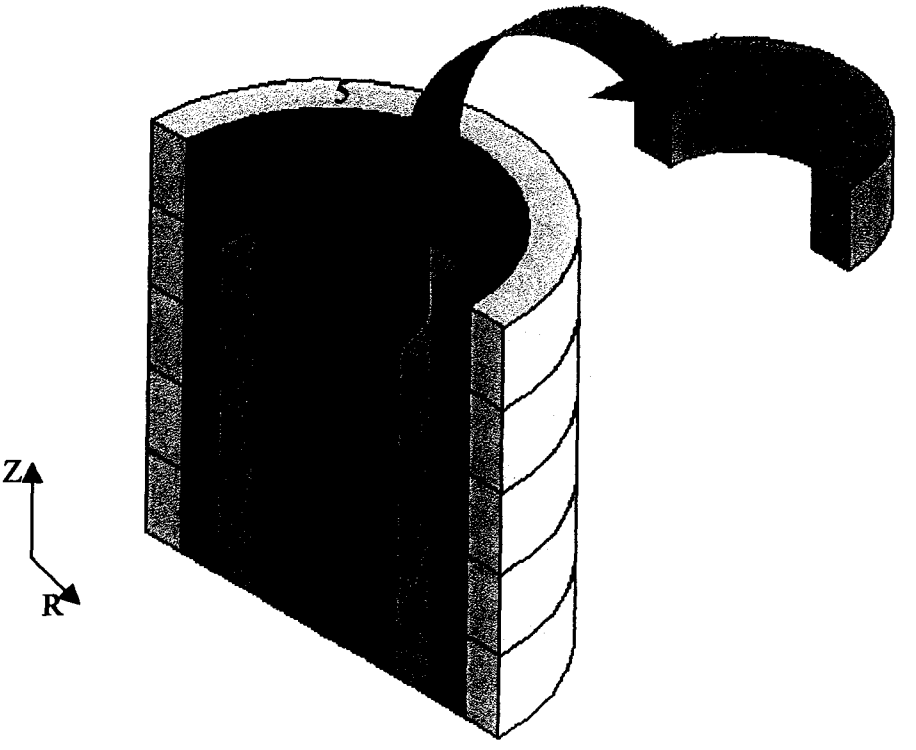


Figure 4.2 Cylindrical Sector from a Reactor Discretized into Finite Volumes

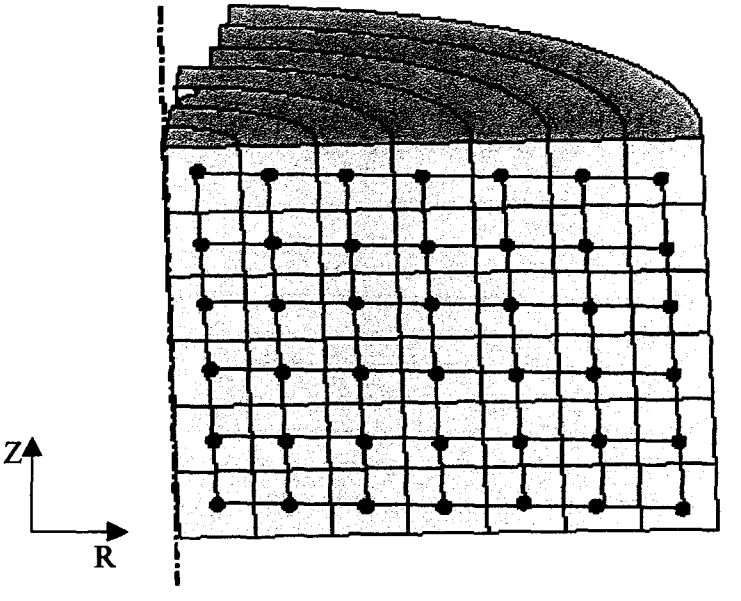


Figure 4.3 Grid Mesh Formation from Cylindrical Sectors

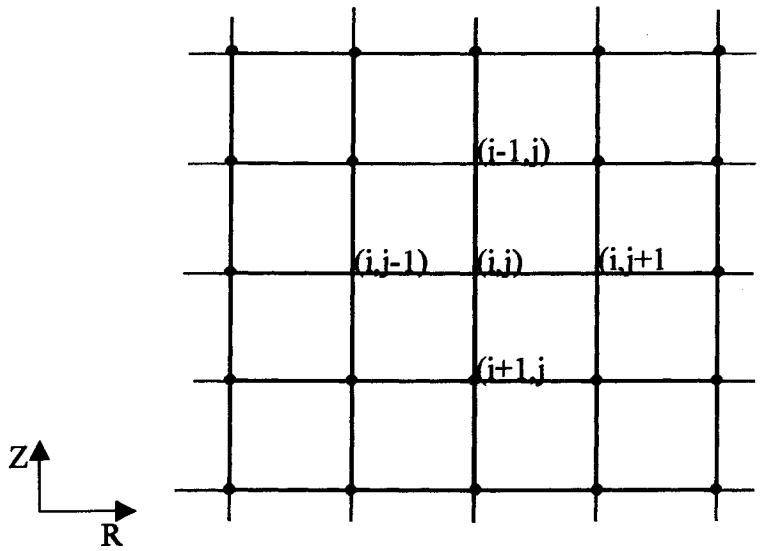


Figure 4.4 Two-Dimensional Grid Mesh

4.2 Implicit Finite Difference Form of the Neutron Diffusion Equations

A simple three by three mesh is considered for the purpose of illustration as shown in the figure 4.5. In a 2-D finite difference model, there are nine types of nodal equations corresponding to the interior nodes, the four sides, and the four corners. In the present work, nodal equations were formed throughout the cylinder volume. As explained in the previous chapter, vacuum boundaries were assumed at the outer radius, and at the top and bottom of the cylindrical reactor.

The general form of the diffusion equation in finite difference form is:

$$\frac{1}{\upsilon}\frac{\phi_i^{t+1}-\phi_i^{t}}{\Delta t} = D\frac{1}{r_{i,j}}\left(\frac{\phi_{i+1,j}^{t+1}-\phi_{i-1,j}^{t+1}}{2\Delta r}\right)+\frac{\phi_{i+1,j}^{t+1}+\phi_{i-1,j}^{t+1}-2\phi_{i,j}^{t+1}}{(\Delta r)^2}+\frac{\phi_{i,j+1}^{t+1}-\phi_{i,j-1}^{t+1}}{(\Delta z)^2}$$
$$-\Sigma_a\phi_i^{t+1}+(1-\beta)\nu\Sigma_f\phi_i^{t+1}+\sum_{i=1}^{6}\lambda_i C^t{}_i+S_i^t \qquad (4.1)$$

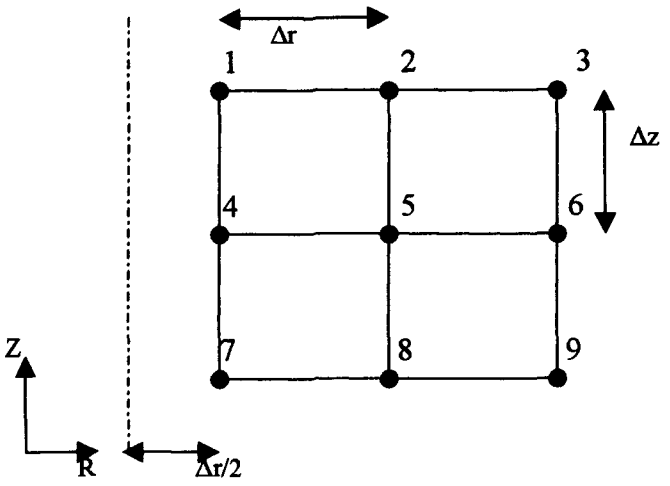


Figure 4.5 Three by Three Nodal Mesh

The following are the finite difference equations applied to nodes within the grid system given in figure 4.5.

Node 1 (Top left corner node)

$$\frac{1}{\upsilon}\frac{\phi_1^{t+1}-\phi_1^{t}}{\Delta t}=D\left[\frac{1}{r_1}\left(\frac{\phi_2^{t+1}-\phi_1^{t+1}}{2\Delta r}\right)+\frac{\phi_2^{t+1}+\phi_1^{t+1}-2\phi_1^{t+1}}{(\Delta r)^2}\right]+D\left[\frac{\phi_4^{t+1}+0-2\phi_1^{t+1}}{(\Delta z)^2}\right] \qquad (4.2)$$
$$-\Sigma_a\phi_1^{t+1}+(1-\beta)\nu\Sigma_f\phi_1^{t+1}+C_1^{t}+S_1^{t}$$

$$D_2\phi_2^{t+1}\left[\frac{1}{2r_1\Delta r}+\frac{1}{(\Delta r)^2}\right]+\frac{D_4\phi_4^{t+1}}{(\Delta z)^2}$$
$$+\phi_1^{t+1}\left[-\frac{1}{\upsilon\Delta t}-\frac{D_1}{(\Delta r)^2}-\frac{2D_1}{(\Delta z)^2}-\frac{D_1}{2r_1\Delta r}-\Sigma_a+(1-\beta)\nu\right]=-\frac{\phi_1^{t}}{\upsilon\Delta t}-C_1^{t}-S_1^{t} \qquad (4.3)$$

Node 2 (Top face nodes)

$$\frac{1}{\upsilon}\cdot\frac{\phi_2^{t+1}-\phi_2^{t}}{\Delta t}=D\left[\frac{1}{r_2}\left(\frac{\phi_3^{t+1}-\phi_1^{t+1}}{2\Delta r}\right)+\frac{\phi_3^{t+1}+\phi_1^{t+1}-2\phi_2^{t+1}}{(\Delta r)^2}\right]+D\left[\frac{\phi_5^{t+1}+0-2\phi_2^{t+1}}{(\Delta z)^2}\right] \qquad (4.4)$$
$$-\Sigma_a\phi_2^{t+1}+(1-\beta)\nu\Sigma_f\phi_2^{t+1}+C_2^{t}+S_2^{t}$$

$$D_3\phi_3^{t+1}\left[\frac{1}{2r_2\Delta r}+\frac{1}{(\Delta r)^2}\right]+\frac{D_5\phi_5^{t+1}}{(\Delta z)^2}+D_1\phi_1^{t+1}\left[\frac{1}{(\Delta r)^2}-\frac{1}{2r_2\Delta r}\right]$$
$$+\phi_2^{t+1}\left[-\frac{1}{\upsilon\Delta t}-\frac{2D_2}{(\Delta r)^2}-\Sigma_a+(1-\beta)\nu\Sigma_f\right]=-\frac{\phi_2^{t}}{\upsilon\Delta t}-C_2^{t}-S_2^{t} \quad (4.5)$$

Node 3 (Top right corner node)

$$\frac{1}{\upsilon}\cdot\frac{\phi_3^{t+1}-\phi_3^{t}}{\Delta t}=D\left[\frac{1}{r_3}\left(\frac{0-\phi_2^{t+1}}{2\Delta r}\right)+\frac{0+\phi_2^{t+1}-2\phi_3^{t+1}}{(\Delta r)^2}\right]+D\left[\frac{0+\phi_6^{t+1}-2\phi_3^{t+1}}{(\Delta z)^2}\right] \quad (4.6)$$
$$-\Sigma_a\phi_3^{t+1}+(1-\beta)\nu\Sigma_f\phi_3^{t+1}+C_3^{t}+S_3^{t}$$

$$D_2\phi_2^{t+1}\left[-\frac{1}{2r_3\Delta r}+\frac{1}{(\Delta r)^2}\right]+\frac{D_6\phi_6^{t+1}}{(\Delta z)^2}+\phi_3^{t+1}\left[-\frac{1}{\upsilon\Delta t}-\frac{2D_3}{(\Delta z)^2}-\Sigma_a+(1-\beta)\nu\cdot\Sigma_f\right] \quad (4.7)$$
$$=-\frac{\phi_3^{t}}{\upsilon\Delta t}-C_3^{t}-S_3^{t}$$

Node 4 (Left face nodes)

$$\frac{1}{\upsilon}\frac{\phi_4^{t+1}-\phi_4^{t}}{\Delta t}=D\left[\frac{1}{r_4}\left(\frac{\phi_5^{t+1}-\phi_4^{t+1}}{2\Delta r}\right)+\frac{\phi_5^{t+1}+\phi_4^{t+1}-2\phi_4^{t+1}}{(\Delta r)^2}\right]+D\left[\frac{\phi_1^{t+1}+\phi_7^{t+1}-2\phi_4^{t+1}}{(\Delta z)^2}\right] \quad (4.8)$$
$$-\Sigma_a\phi_4^{t+1}+(1-\beta)\nu\Sigma_f\phi_4^{t+1}+C_4^{t}+S_4^{t}$$

$$D_5\phi_5^{t+1}\left[\frac{1}{(\Delta r)^2}+\frac{1}{2r_4\Delta r}\right]+\frac{D_1\phi_1^{t+1}}{(\Delta z)^2}+\frac{D_7\phi_7^{t+1}}{(\Delta z)^2}$$
$$+\phi_4^{t+1}\left[-\frac{1}{\upsilon\Delta t}-\frac{D_4}{(\Delta r)^2}-\frac{2D_4}{(\Delta z)^2}-\frac{D_4}{2r_4\Delta r}-\Sigma_a+(1-\beta)\nu\Sigma_f\right]=-\frac{\phi_4^{t}}{\upsilon\Delta t}-C_4^{t}-S_4^{t} \quad (4.9)$$

Node 5 (Central nodes)

$$\frac{1}{\upsilon}\frac{\phi_5^{t+1}-\phi_5^{t}}{\Delta t}=D\left[\frac{1}{r_5}\left(\frac{\phi_6^{t+1}-\phi_4^{t+1}}{2\Delta r}\right)+\frac{\phi_6^{t+1}+\phi_4^{t+1}-2\phi_5^{t+1}}{(\Delta r)^2}\right]+D\left[\frac{\phi_2^{t+1}+\phi_8^{t+1}-2\phi_5^{t+1}}{(\Delta z)^2}\right] \quad (4.10)$$
$$-\Sigma_a\phi_5^{t+1}+(1-\beta)\,\nu\,\Sigma_f\phi_5^{t+1}+C_5^{t}+S_5^{t}$$

$$D_6\phi_6^{t+1}\left[\frac{1}{2r_5\Delta r}+\frac{1}{(\Delta r)^2}\right]+D_4\phi_4^{t+1}\left[\frac{1}{(\Delta r)^2}-\frac{1}{2r_5\Delta r}\right]+\frac{D_2\phi_2^{t+1}}{(\Delta z)^2}+\frac{\phi_8^{t+1}}{(\Delta z)^2}$$
$$+\phi_5^{t+1}\left[-\frac{1}{\upsilon\Delta t}-\frac{2D_5}{(\Delta r)^2}-\frac{2D_5}{(\Delta z)^2}-\Sigma_a+(1-\beta)\,\nu\,\Sigma_f\right]=-\frac{\phi_5^{t}}{\upsilon\Delta t}-C_5^{t}-S_5^{t} \tag{4.11}$$

Node 6 (Right face nodes)

$$\frac{1}{\upsilon}\frac{\phi_6^{t+1}-\phi_6^{t}}{\Delta t}=D\left[\frac{1}{r_6}\left(\frac{0-\phi_5^{t+1}}{2\Delta r}\right)+\frac{0+\phi_5^{t+1}-2\phi_6^{t+1}}{(\Delta r)^2}\right]+D\left[\frac{\phi_3^{t+1}+\phi_9^{t+1}-2\phi_6^{t+1}}{(\Delta z)^2}\right] \tag{4.12}$$
$$-\Sigma_a\phi_6^{t+1}+(1-\beta)\,\nu\,\Sigma_f\phi_6^{t+1}+C_6^{t}+S_6^{t}$$

$$D_5\phi_5^{t+1}\left[\frac{1}{2r_6\Delta r}+\frac{1}{(\Delta r)^2}\right]+\frac{D_3\phi_3^{t+1}}{(\Delta z)^2}+\frac{D_9\phi_9^{t+1}}{(\Delta z)^2}$$
$$+\phi_6^{t+1}\left[-\frac{1}{\upsilon\Delta t}-\frac{2D_6}{(\Delta r)^2}-\frac{2D_6}{(\Delta z)^2}-\Sigma_a+(1-\beta)\,\nu\,\Sigma_f\right]=-\frac{\phi_6^{t}}{\upsilon\Delta t}-C_6^{t}-S_6^{t} \tag{4.13}$$

Node 7 (Bottom left corner node)

$$\frac{1}{\upsilon}\frac{\phi_7^{t+1}-\phi_7^{t}}{\Delta t}=D\left[\frac{1}{r_7}\left(\frac{\phi_8^{t+1}-\phi_7^{t+1}}{2\Delta r}\right)+\frac{\phi_8^{t+1}+\phi_7^{t+1}-2\phi_7^{t+1}}{(\Delta r)^2}\right]+D\left[\frac{0+\phi_4^{t+1}-2\phi_7^{t+1}}{(\Delta z)^2}\right] \tag{4.14}$$
$$-\Sigma_a\phi_1^{t+1}+(1-\beta)\nu\Sigma_f\phi_7^{t+1}+C_7^{t}+S_7^{t}$$

$$D_8\phi_8^{t+1}\left[\frac{1}{(\Delta r)^2}+\frac{1}{2r_7\Delta r}\right]+\frac{D_4\phi_4^{t+1}}{(\Delta z)^2}$$
$$+\phi_7^{t+1}\left[-\frac{1}{\upsilon\Delta t}-\frac{D_7^{t+1}}{(\Delta r)^2}-\frac{2D_7^{t+1}}{(\Delta z)^2}-\frac{D_7^{t+1}}{2r_7\Delta r}-\Sigma_a+(1-\beta)\nu\Sigma_f\right]=-\frac{\phi_7^{t}}{\upsilon\Delta t}-C_7^{t}-S_7^{t} \tag{4.15}$$

Node 8 (Bottom face nodes)

$$\frac{1}{\upsilon}\frac{\phi_8^{t+1}-\phi_8^{t}}{\Delta t}=D\left[\frac{1}{r_8}\left(\frac{\phi_9^{t+1}-\phi_7^{t+1}}{2\Delta r}\right)+\frac{\phi_9^{t+1}+\phi_7^{t+1}-2\phi_8^{t+1}}{(\Delta r)^2}\right]+D\left[\frac{0+\phi_5^{t+1}-2\phi_8^{t+1}}{(\Delta z)^2}\right] \tag{4.16}$$
$$-\Sigma_a\phi_8^{t+1}+(1-\beta)\,\nu\,\Sigma_f\phi_8^{t+1}+C_8^{t}+S_8^{t}$$

$$D_9\phi_9^{t+1}\left[\frac{1}{2r_8\Delta r}+\frac{1}{(\Delta r)^2}\right]+D_7\phi_7^{t+1}\left[\frac{1}{(\Delta r)^2}-\frac{1}{2r_8\Delta r}\right]+\frac{D_5\phi_5^{t+1}}{(\Delta z)^2}+$$
$$\phi_8^{t+1}\left[-\frac{1}{\upsilon\Delta t}-\frac{2D_8^{t+1}}{(\Delta r)^2}-\frac{2D_8^{t+1}}{(\Delta r)^2}-\Sigma_a+(1-\beta)\,\nu\,\Sigma_f\right]=-\frac{\phi_8^{t}}{\upsilon\Delta t}-C_8^{t}-S_8^{t} \quad (4.17)$$

Node 9 (Bottom right corner node)

$$\frac{1}{\upsilon}\frac{\phi_9^{t+1}-\phi_9^{t}}{\Delta t}=D\left[\frac{1}{r_9}\left(\frac{0-\phi_8^{t+1}}{2\Delta r}\right)+\frac{0+\phi_8^{t+1}-2\phi_9^{t+1}}{(\Delta r)^2}\right]+D\left[\frac{\phi_{i+j}^{t+1}+\phi_6^{t+1}-2\phi_9^{t+1}}{(\Delta z)^2}\right] \quad (4.18)$$
$$-\Sigma_a\phi_9^{t+1}+(1-\beta)\,\nu\,\Sigma_f\phi_9^{t+1}+C_9^{t}+S_9^{t}$$

$$D_8\phi_8^{t+1}\left[-\frac{1}{2r_9\Delta r}+\frac{1}{(\Delta r)^2}\right]+\frac{D_6\phi_6^{t+1}}{(\Delta z)^2}+$$
$$\phi_9^{t+1}\left[-\frac{1}{\upsilon\Delta t}-\frac{2D_9^{t+1}}{(\Delta r)^2}-\frac{2D_9^{t+1}}{(\Delta z)^2}-\Sigma_a+(1-\beta)\,\nu\,\Sigma_f\right]=-\frac{\phi_9^{t}}{\upsilon\Delta t}-C_9^{t}-S_9^{t} \quad (4.19)$$

Once these nodal equations were defined for all nodes, the resulting simultaneous equations were solved for the flux at each timestep by arranging them in matrix form as

$$\left[\mathbf{A}\right]_{N\times N}\left[\boldsymbol{\varphi}^{t+1}\right]_{N\times 1}=-\frac{1}{\upsilon\Delta t}\left[\boldsymbol{\varphi}^{t}\right]_{N\times 1}-\left[\mathbf{S}^{t}\right]_{N\times 1}-\left[\mathbf{C}^{t}\right]_{N\times 1} \quad (4.20)$$

where, N is the number of finite volumes within the reactor

υ is the neutron speed

Δt is timestep

A is coefficient matrix

ϕ^t, ϕ^{t+1} are the matrices describing the neutron flux in the present and future time steps

C^t is the matrix representing delayed neutrons in the present time step

S^t is the matrix representing the source neutrons from the accelerator

The coefficient matrix is diagonally dominant. As can be seen from equation 4.20, in each timestep of the reactor simulation, solution of the simultaneous equations requires the inversion of the coefficient matrix, A. The Gauss-Seidel iterative method was used to invert the matrix.

4.3 Gauss-Seidel Iterative Method[48]

The Gauss-Seidel iterative method is one of the simplest and most effective methods for solving a system of equations. The procedure in solving a general system of equations is to: 1) make initial guesses for all unknown values of the dependent variable; 2) solve each equation for the dependent variable using the initial guesses and the most recently computed values; 3) repeat the solution of the equations in this manner until changes in the unknowns become smaller than a prescribed tolerance.

The initial guesses used at each time step consist of the results from the previous timestep. The general algorithm for Gauss-Seidel iteration is given as

$$x_i = \frac{1}{a_{ii}}\left(c_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j\right), \qquad i = 1,2,....,n \tag{4.21}$$

where the x-values on the right hand side are the most recently computed values for x.

The equations must be organized so that the coefficient matrix is diagonally dominant, as given by:

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \qquad i = 1,2,\ldots,n \tag{4.22}$$

and

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \qquad \textit{for at least one } i \tag{4.23}$$

Though this is a sufficient condition for convergence to the solution, convergence may sometimes be observed even when it is not met.

The Gauss-Siedel method varies from the Jacobi iteration method where values of x will be updated only after calculating all (n) variables in each iteration. By using values that are continuously updated during the solution, the Gauss-Seidel method converges more rapidly.

CHAPTER 5

SIMULATION CODE

To study the time-dependent behavior of a nuclear reactor coupled to a particle accelerator, the transmuter simulation code "ADSTRANS" was developed. ADSTRANS relies upon MCNPX, a Monte Carlo particle transport code developed by the Los Alamos National Laboratory, to generate neutron production due to the accelerator. MCNPX was also used to determine the effective neutron multiplication factor, (k_{eff}), of the reactor blanket. In this chapter, the organization of the simulation code ADSTRANS as well as the fundamentals of MCNPX are discussed.

"ADSTRANS" is a finite difference neutron transport code written in programming language ANSI C. C combines the features of a high-level language with excellent portability between UNIX, LINUX, and Windows-based systems.[49] "C stands out among general-purpose programming languages for its unrivaled mix of portability, power, flexibility, and elegance. Because it compiles to highly efficient machine code, it is particularly well-suited to scientific and engineering applications."[50]

5.1 Organization of ADSTRANS

An overview of the organization of "ADSTRANS" is presented through a flow chart as shown in figure 5.1. Further details of the organization of the code are presented in the latter part of this chapter.

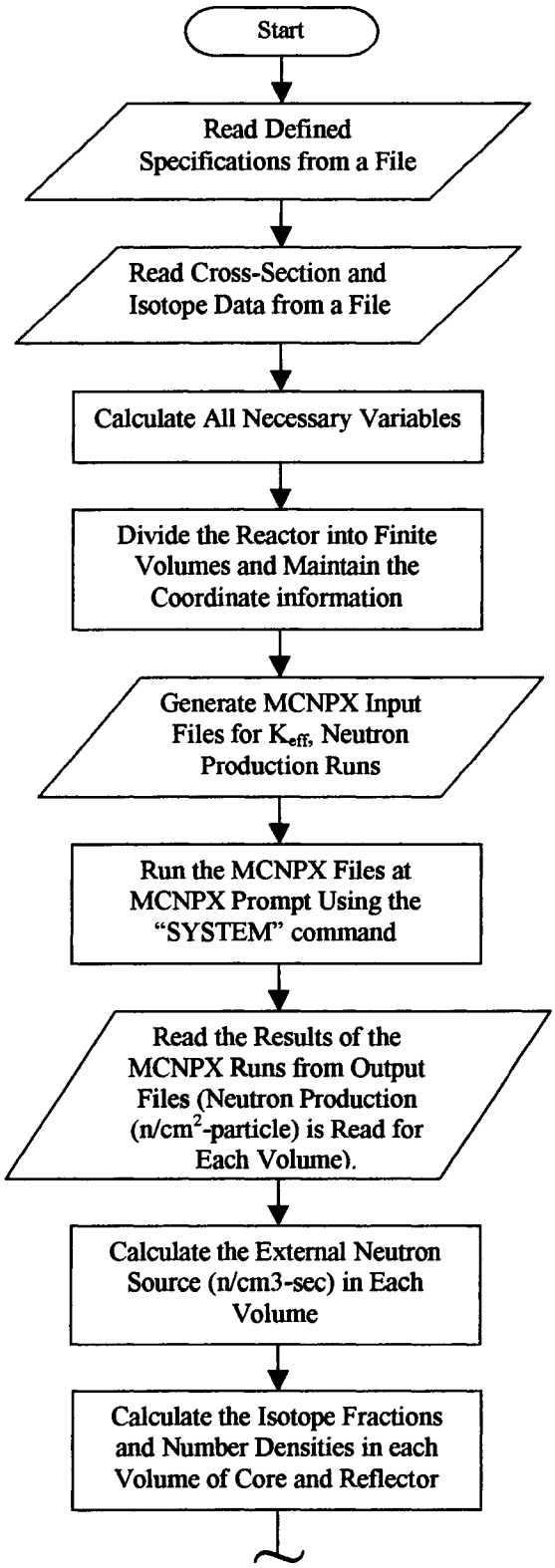


Figure 5.1 Flowchart Describing the Organization of the ADSTRANS Code

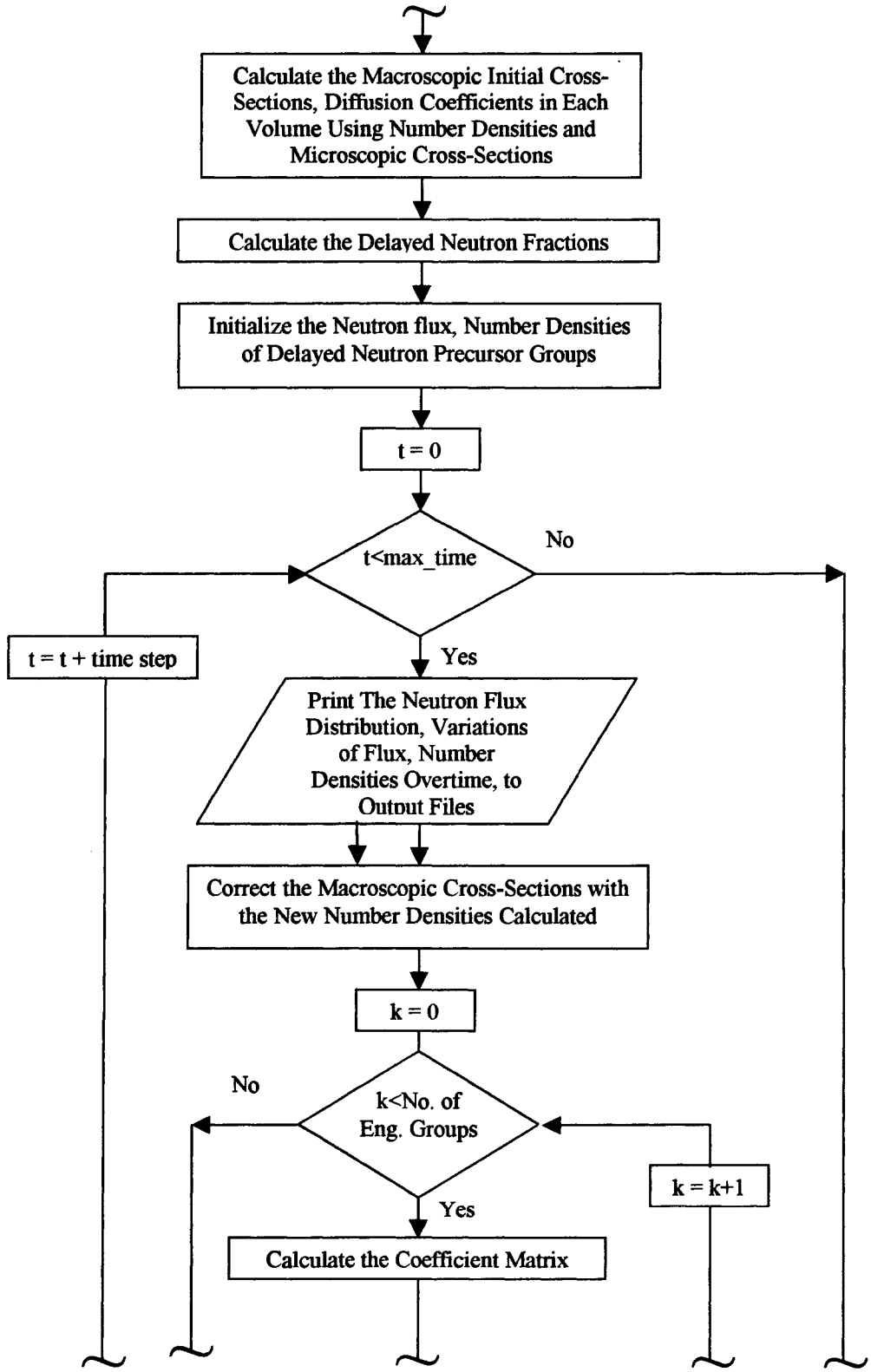


Figure 5.1 (Continued.....)

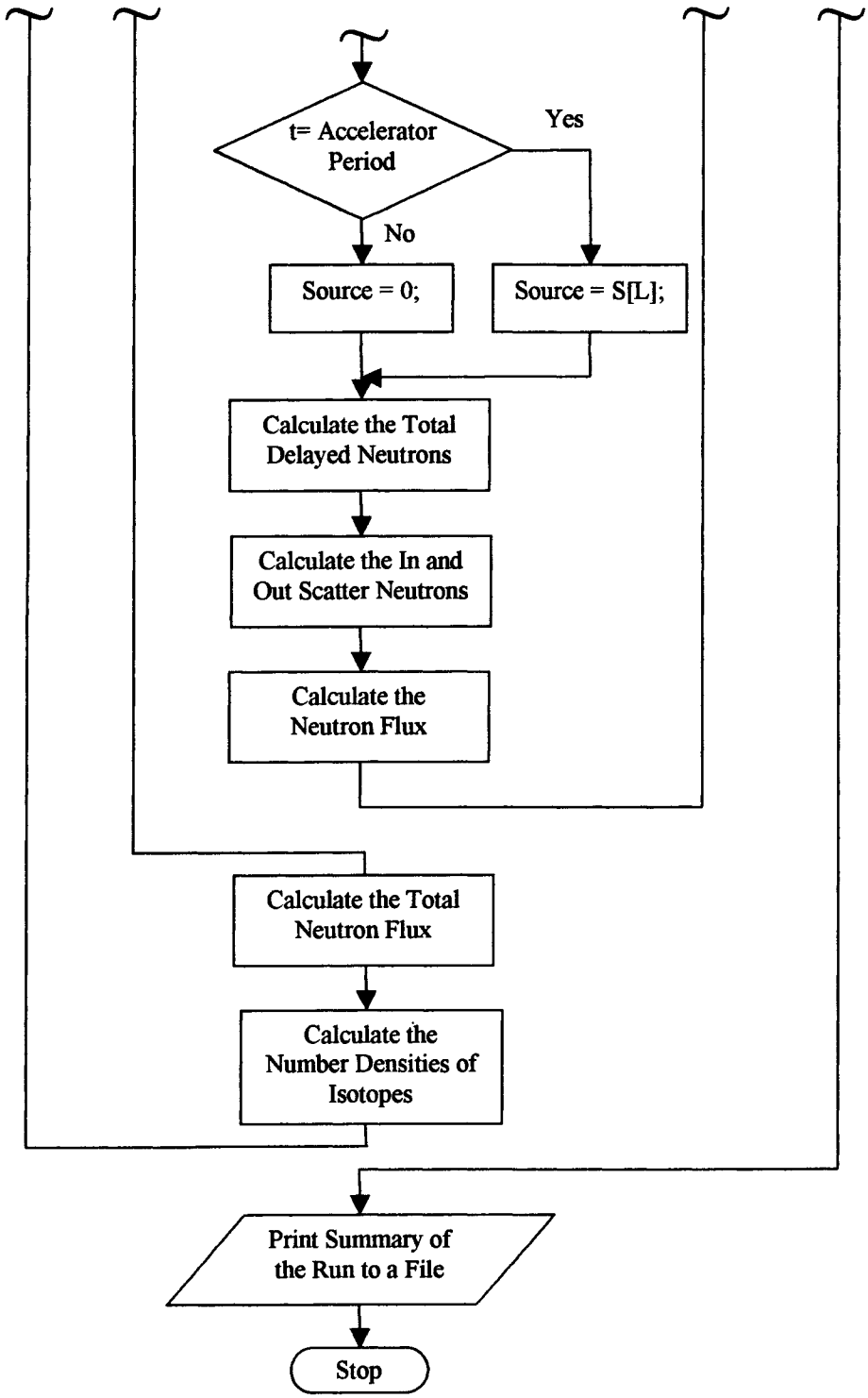


Figure 5.1 (Continued)

5.1.1 Input the Required Data

As shown in the figure 5.1, the ADSTRANS program initializes by reading the user defined details of the reactor (reactor dimensions, mesh size, etc.), accelerator (particle type, energy, etc.) through a subroutine named "get_setupdata". The program will also read the nuclear cross-sections, atomic numbers, and atomic weights from a separate nuclear data file called "nuclear.dat" through a subroutine named "get_properties." The "nuclear.dat" file contains the cross-sections and other basic information for all of the isotopes that are used in this analysis. The cross-section data was obtained from ENDF/B libraries using MCNPX to print or plot the cross-sections.

5.1.2 Calculate Material Properties

Properties, such as absorption cross-sections, are calculated using radiative capture and fission cross-sections read from the data file. Other important properties, such as the weight percent of different isotopes in the core and in the reflector are calculated using the data read from the data files and predefined data within the program.

The reactor was divided into finite volumes\cells through a subroutine named "cell_dimensions". Coordinate information and the radial and axial dimensions of each cell were calculated and read into the appropriate variables.

5.1.3 MCNPX Input File Generation

In the simulation software, the particle accelerator coupled to the nuclear reactor generated source neutrons in the reactor and these neutrons were multiplied in the reactor core. These neutrons were then responsible for the transmutation of radioisotopes present

in the reactor. The source neutrons produced in the reactor by the accelerator and the effective neutron multiplication due to fission, k_{eff}, were simulated by the Monte Carlo N-Particle transport code, MCNPX. Details of MCNPX are presented in the latter part of this chapter.

In the simulation, a high energy particle beam from the accelerator impinged upon a high-Z material target made of tantalum, tungsten, lead, or bismuth. This resulted in the production of neutrons through spallation, (p,n), or photodisintegration, (γ,n). The ADSTRANS program automatically generated two separate MCNPX input files based on the user-defined reactor geometry. ADSTRANS ran MCNPX as a separate process using the output to determine the neutron source term from the accelerator. This source term generated by the subroutine "make_mcnpx_neutron" was then used during the simulation to produce neutrons during the time that the accelerator was firing. A second MCNPX job using the subroutine "make_mcnpx_keff" then was run for the same geometry to verify the effective neutron multiplication factor for the reactor, k_{eff}. ADSTRANS used cell coordinate information, isotope fractions and other required information to generate the input files. The MCNPX results were written to disk in files named "keff" and "neutron." The ADSTRANS program extracted neutron flux per cell and the value of k_{eff} directly from the MCNPX files, although the user may investigate these MCNPX files directly, if desired. Sample MCNPX input files generated by ADSTRANS are presented in the Appendix V.

5.1.4 Execute the MCNPX Simulations

Once the input files are created they were run as an external process through a batch file, "mcnpx.bat," using the "system" command in C. This command allows any operating system command to be run as an external process. Control returns to the C program once the external process completes.

system ("mcnpx.bat")

The batch file "mcnpx.bat" contained the commands to: 1) delete all files that were created in the previous runs, 2) change to the MCNPX binary director and set any environment variables for MCNPX, if required, 3) run the input files as separate, external processes with specific output filenames (e.g. "mcnpx i=keff –o keff"), and, 4) return to the C program to continue the ADSTRANS simulation. The geometry of the reactor and the accelerator target in these simulations could be checked through the geometric plots generated by MCNPX. A ten by ten mesh of the cylindrical geometry of the reactor is shown in figure 5.2.

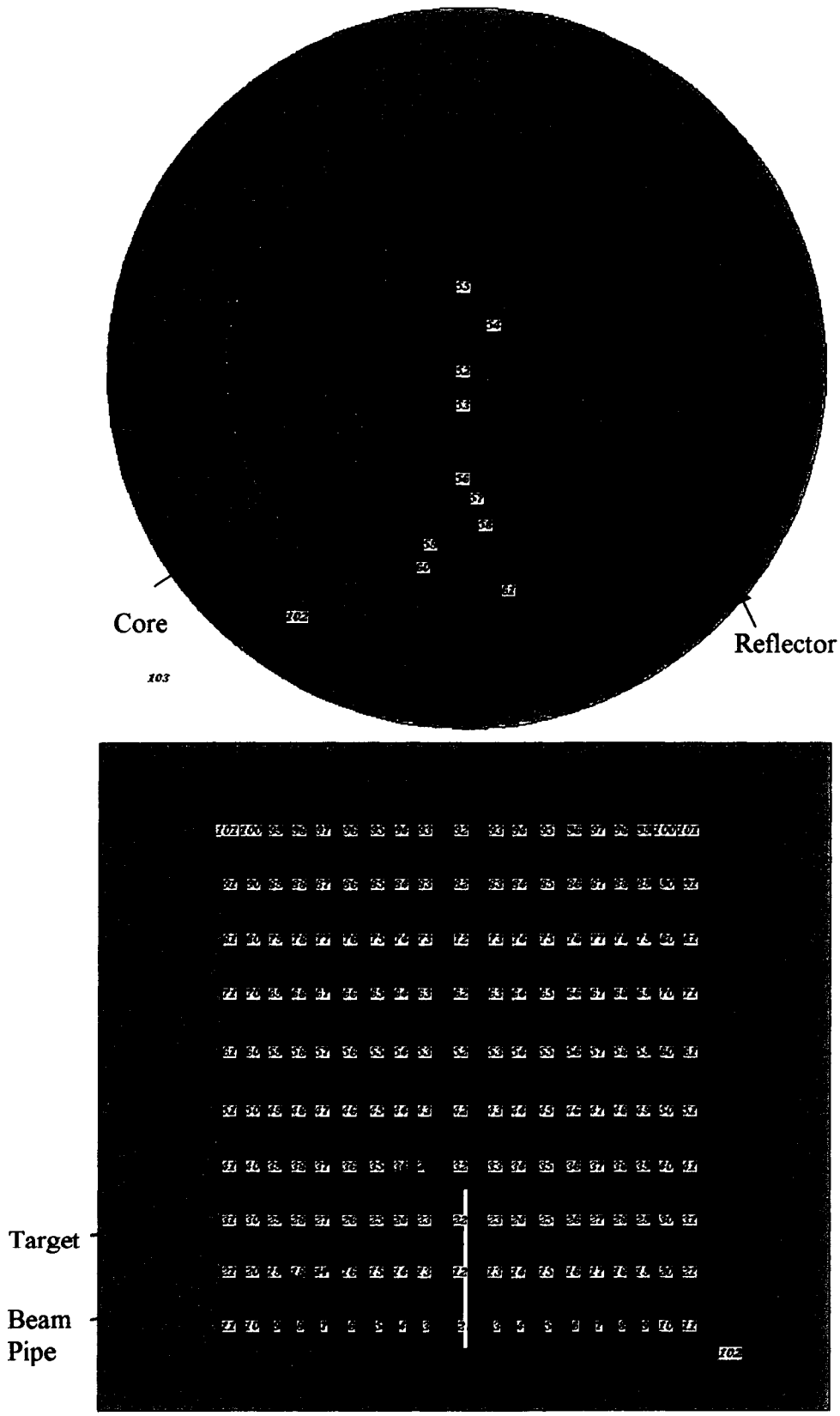


Figure 5.2 Geometric Plot of the Reactor with the Target, Generated by MCNPX

5.1.5 Read the MCNPX Output

Once the MCNPX simulations were complete, MCNPX generated output files of the simulations containing detailed results. The MCNPX code wrote it's output files in a consistent manner, so keywords for important results were searched for in the ADSTRANS code using C string commands. Two ADSTRANS subroutines, "Read_keff" and "Read_Neutron," read the information from the MCNPX output files and into the prescribed variables. The neutron production as a function of neutron energy was read from the MCNPX output file for each cell within the mesh that described the reactor. Care was taken to use the same mesh for both the MCNPX simulations and the subsequent ADSTRANS calculations. The effective multiplication factor, k_{eff}, was also read from MCNPX output to verify the criticality of the reactor defined by the user.

5.1.6 Calculate the Neutron Production

The neutron production read from the MCNPX output listed the number of neutrons produced per accelerator particle as a function of neutron energy and cell location. The user was allowed to enter data on the accelerator characteristics. These included the type of accelerator used (electron or proton), the emitted particle energy, the firing rate in pulses per second, the portion of each cycle that the accelerator is "on," and the current of the particles emitted each time that the accelerator fired. The total neutron production during each firing could then be calculated. The following equations were used to calculate the subsequent neutron flux for each cell at each energy group due to the particle accelerator.

Total number of particles in each pulse:

$$particles\ per\ pulse = \frac{accelerator\ pulse\ charge}{particle\ charge}$$

Source neutron fluence:

$$neutron\ fluence = neutron\ fluence\ per\ particle[L][k] \times particles\ per\ pulse,$$

Source neutron flux:

$$source\ flux[L][k] = neutron\ fluence[L][k] \times accelerator\ frequency.$$

where L is cell index and k is energy group index, this notation is followed through out this program.

5.1.7 Calculate Initial Number Densities

The average density of each isotope within the reactor is calculated using their weight percentage. The number density of each isotope, defined in units of atoms per unit volume, for each cell in the reactor is then calculated using the atomic weight and the average density in the reactor as shown in the following example.

$$N\_U238[L] = \frac{average\ density\ of\ U238\ in\ the\ reactor \times Avagadros\ number}{atomic\ weight\ of\ U238}.$$

Macroscopic cross-sections in each cell at each energy group are calculated from the number densities and their microscopic cross-sections as shown in the following example.

$$Sigma\ scatter[L][k] = N\_H[L] \times H\_sigma\ scatter[k] + N\_O[L] \times O\_sigma\ scatter[k] \\ N\_U238[L] \times U238\_sigma\ scatter[k] + ...;$$

where the variable 'Sigma scatter' on the left hand side is the macroscopic scattering cross-section in each cell at each energy group and 'sigma scatter' on the right hand side

is the microscopic scattering cross-section of the isotope. The neutron diffusion coefficient in each cell at each energy group is calculated from the macroscopic transport cross-section from the formula given in Chapter 3.

5.1.8 Calculate the Delayed Neutron Fraction

Delayed neutrons are created by the radioactive decay of fission products in a reactor. As opposed to fission neutrons that are generated during fission, delayed neutrons may be generated many seconds after the fission products are created. Delayed neutrons have an importance in reactor behavior since they extend the average neutron lifetime and serve to increase the reactor period. This helps in the control of a reactor.

The delayed neutrons fraction in each cell for each delayed group will be calculated from delayed neutron fraction (β) of each delayed group and the macroscopic fission cross-section of each isotope in each cell as shown in the following example.

$$delayed\ source[L][i][k] = beta[L][i][k] \times \begin{pmatrix} U235\_mu[k] \times N\_U235[L] \times U235.sigma\_f[k] \\ + Pu239\_mu[k] \times N\_Pu239[L] \times Pu239.sigma\_f[k] + ... \end{pmatrix}$$

where the new index variable *i* is the delayed neutron group index, *nu* is the average neutron production in each fission reaction and *sigma_f* is the fission cross-section.

Initializing the variables at zero time

The neutron flux, the total delayed neutron population, and the concentration of neutron poisons in the reactor are initialized to zero at time zero when the simulation begins.

5.1.9 Start the Time Iteration

Once all variables describing the reactor and accelerator behavior are defined, the ADSTRANS program begins computing the neutron distribution as a function of time and position within the reactor. The calculations are carried out through finite time steps of 0.1 milliseconds or less. The iterations over time start at reactor time zero and continue until the pre-specified maximum time is reached. In each iteration, the code will: 1) write all the important results to output files, 2) calculate the variations in the number densities of important isotopes which can change rapidly and effect the reactor behavior, 3) correct the macroscopic cross-sections with the varying number densities, 4) calculate coefficient matrix with the most recently calculated variables, 5) read source neutrons from accelerator production based on test conditions and MCNPX output, 6) calculate the total delayed neutron population, fuel burnup, and neutron poison accumulation, 7) calculate the neutron flux into and out of each energy group, and 8) calculate the neutron flux at each timestep. Finally, terminate the program by writing a file with the summary of the simulation results.

5.1.10 Write Results to Output Files

The results of the analysis, such as the neutron flux variation inside the reactor, depletion of the isotopes in the reactor, and buildup of neutron poisons are written to separate files ("flux.out," "Pu.out," etc). The neutron flux, depletion of isotopes and neutron poisons at each node in the reactor are also written to separate files ("Nodeflux.OUT", "PuNode") at regular intervals of time to monitor variations of these over time.

5.1.11 Correcting the Macroscopic Cross-Sections

The macroscopic cross-sections inside the reactor vary over reactor operation, due to the variations in the number densities of the isotopes in the reactor. Hence, in each time step these cross-sections are corrected using the current number densities.

5.1.12 Calculate the Coefficient Matrix

To solve the neutron transport equations in the finite difference form within the reactor, separate difference equations are created at each node for each energy group. For three energy groups and a 20 x 20 mesh, 1200 simultaneous equations must be solved at each timestep for the resulting neutron flux. For M radial nodes, N axial nodes, and G energy groups, a square coefficient matrix with M x N rows is formed and G number of inversions of coefficient matrix are done in each timestep. The coefficient matrix is calculated in each timestep with the most recently calculated values for number densities, cross-sections, and boundary conditions. The elements of the coefficient matrix are calculated from the equations defined for the nine unique kinds of nodes found in a 2-D system.

In each iteration, solutions of the transport equations require the inversion of the coefficient matrix. The speed of inversion depends upon the mesh density and, hence, the size of the matrix. Inversion of this coefficient matrix in each time step requires most of the computer time required to run the program. A modified Gauss-Seidel algorithm was used to invert the coefficient matrix and matrix subroutines in C were used to create the final array that described the neutron flux at each time step. The modified Gauss-Seidel routine used the results of the previous time step to provide seed values for

the next time step. The routine also minimized execution speeds by only storing diagonal and non-zero off-diagonal components in five vectors of length M x N. This increased execution speed by two to three orders of magnitude.

5.1.13 Accelerator Production of Source Neutrons

Generally, accelerators are pulsed at a regular frequency (e.g. 60 Hz). Hence, the neutron production from an accelerator is typically not continuous. When the accelerator fires, protons or electrons strike a heavy metal target deep within the reactor generating neutrons through spallation, (p,n), or photodisintegration, (γ,n). The accelerator serves as a source of neutrons to assist the neutron population generated by fission within the reactor. This source term exists only when the accelerator is fired and it decreases to zero between firings. As an example, the electron accelerator at the Idaho Accelerator Center to test neutron production had a frequency of 60 Hz and the "on" time was 2 microseconds. In this case, the accelerator was only on for about 2 μs / 16.67 ms or 0.01% of each cycle.

To account for the pulse rate and duty cycle of the accelerator, the external source term in the transport equation in ADTRANS was read through a subroutine named "source" which distinguished between the on and off conditions of the accelerator. This subroutine contained the information on the accelerator period between each firing and the "on" time during each cycle. This neutron source term was then applied in the numerical solution of the neutron transport equations within the reactor.

5.1.14 Total Delayed Neutron Population

Using the fission fractions of each delayed neutron precursor group and their decay constants, the number densities of each precursor group were calculated in each time step. Once the number densities are calculated, the neutron production from each precursor group along with the total neutron production was calculated.

5.1.15 Calculate Neutron Scattering Between Energy Groups

Using the macroscopic group transfer cross-sections and the neutron flux in each energy group, neutron flux due to scattering collisions into and out of each energy group was calculated. In this analysis, only down-scattering from high energy to low energy groups was considered. The energy groups were "directly connected," where neutrons were allowed to scatter from one energy group into the next lower energy group only.

5.1.16 Calculate the Neutron Flux

Once all variables in the transport equation are calculated, the neutron flux in each timestep was calculated from the flux in the previous time step by inverting the coefficient matrix. The matrix inversion is done by the Gauss-Seidel method as explained earlier. Once the maximum reactor time specified by the user was reached, the iterations were terminated.

5.1.17 Summary Report on the Simulation

After all time iterations were completed, a summary report of the simulation was printed to an output file named "Reslults.out." Important input specifications were

presented along with calculated variables and assumptions used in the code. Execution time for the simulation was also written to this file. A sample summary file of the simulation is presented in the Appendix IV. The ADTRANS program terminated once the summary of the simulations were written to the output file.

5.2 Monte-Carlo N-Particle Transport Code, MCNPX

The particle beam coming from an accelerator impinged upon a high-Z material target and produced neutrons through photodisintegration, (γ,n), or spallation, (p,n), reactions. The neutrons produced in the target diffused into the surrounding subcritical reactor blanket around the target. The neutron production inside the target and the transport of neutrons in the reactor during accelerator firing were simulated using the Monte-Carlo N-Particle transport code, MCNPX written by the Los Alamos National Laboratory. MCNPX version 4.2j was used in this analysis.

MCNPX is based on MCNP. MCNP is a general purpose, continuous energy, generalized-geometry, time dependent, and coupled neutron/photon/electron transport code. It can be used in several transport neutrons, photons, electrons, and other subatomic particles either individually, or in combination. This version of MCNPX modeled neutron energy from 10^{-11} MeV to 20 MeV, and the photon and electron energy from 1 keV to 1000 MeV. It also has the capability to calculate k_{eff} for fissile systems.[43, 44]

In using MCNPX, users specify details of the problem through an input file written in textual form that is subsequently read by MCNPX. An input file contains the geometry specification, description of materials, location and characteristics of the

neutron, photon, or electron source, type of answers or tallies desired, and variance reduction techniques used to improve efficiency. ADSTRANS automatically developed MCNPX input files with all details based on the user-specified information about the dimensions of the reactor, target, material of the target, type of accelerator etc.

CHAPTER 6

ANALYTICAL METHOD

In this chapter, the different analytical approaches used to solve several kinds of reactor problems are discussed. These analytical solutions were used to verify the ability of the ADSTRANS code to accurately predict transient reactor or transmuter behavior.

6.1 Neutron Flux Distribution in a Finite Bare Cylindrical Reactor

A cylindrical reactor core with height H, radius R as shown in figure 6.1 was considered and the neutron flux in the radial and axial direction was analyzed.

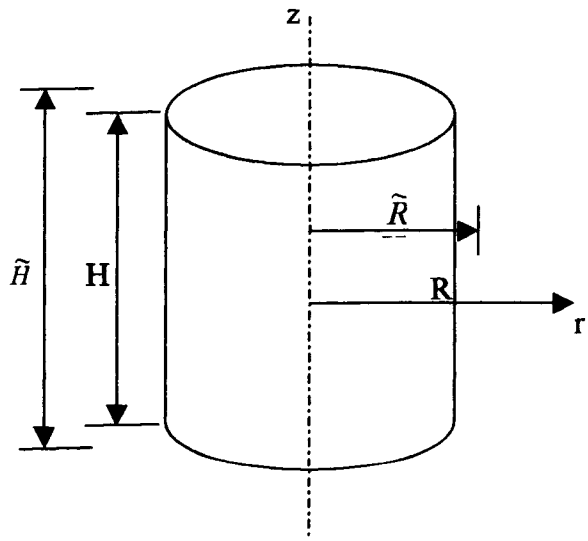


Figure 6.1 Finite Bare Cylindrical Reactor Core

$\tilde{H}, \tilde{R}$ are the extrapolated height and radius of the reactor. The neutron flux at these extrapolated boundaries is assumed to vanish and this serves as a boundary condition to the second-order differential equations that model reactor behavior. The extrapolated height and radius are expressed as

$$\tilde{H} = H + 2d \tag{6.1a}$$

$$\tilde{R} = R + d \tag{6.1b}$$

where d is the extrapolated distance, which is given by

$$d = 0.71\lambda_{tr} \tag{6.2}$$

where λ_{tr} is the transport mean free path of neutrons traveling in the reactor medium. The transport mean free path equals three times the diffusion coefficient D. Hence, the extrapolated distance, d, in terms of diffusion coefficient, D, is

$$d = 2.13D \tag{6.3}$$

The one energy group neutron transport equation for a steady state reactor is

$$\nabla^2\phi + B^2\phi = 0 \tag{6.4}$$

The distribution of neutron flux in a finite cylinder depends upon both radial distance, r, from the central axis and axial distance from the center of the cylinder, z. The general one-group reactor equation for this specific case becomes

$$\frac{1}{r}\frac{\partial}{\partial r}r\frac{\partial\phi}{\partial r} + \frac{\partial^2\phi}{\partial z^2} + B^2\phi = 0 \tag{6.5}$$

or

$$\frac{\partial^2\phi}{\partial r^2} + \frac{1}{r}\frac{\partial\phi}{\partial r} + \frac{\partial^2\phi}{\partial z^2} + B^2\phi = 0 \tag{6.6}$$

and the boundary conditions are $\phi(\tilde{R}, z) = 0$ *and* $\phi(z, \tilde{H}/2) = 0$, neutron flux is zero at extrapolated distances in the radial and axial boundaries. Equation 6.6 is solved through separation of variables, where, $\phi(r, z) = R(r)Z(z)$. The distribution of the neutron flux for the steady-state bare reactor in terms of radial and axial position is

$$\phi(r, z) = AJ_0\left(\frac{2.405r}{\tilde{R}}\right)\cos\left(\frac{\pi z}{\tilde{H}}\right) \tag{6.7}$$

where A is the maximum neutron flux at the center of the reactor.

6.2 Transient Neutron Flux in a Finite Bare Cylindrical Reactor

The numerical reactor simulation code, ADSTRANS, can be verified by comparing it's results to accurate analytical solutions. To verify the transient neutron flux from the simulation code ADSTRANS, a finite bare cylindrical core with specific boundary and initial conditions is considered as shown in the figure 6.2. The core was assumed to be

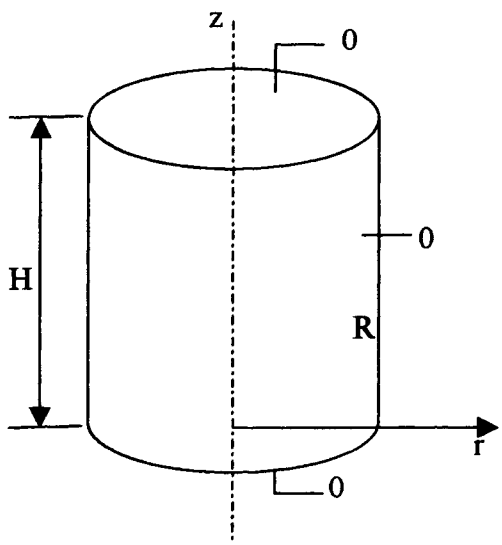


Figure 6.2 Finite Bare Cylindrical Reactor Core with Specific Boundary, Initial Conditions

uniformly distributed with neutron flux ϕ^0, at time t = 0 and the boundaries of the reactor are surrounded by a vacuum, as shown. The core was assumed to be filled with a medium that did not absorb neutrons and did not multiply the neutron population, in other words, a pure neutron diffusion case was considered. The one group, time dependent diffusion equation is

$$D\nabla^2\phi + B^2\phi = \frac{1}{\upsilon}\frac{\partial\phi}{\partial t} \tag{6.8}$$

where B is reactor buckling and is given as

$$B = \frac{1}{D}\left(\frac{1}{k}\Sigma_f - \Sigma_a\right) \tag{6.9}$$

For a non-absorbing medium, the macroscopic absorption cross-section, Σ_a is zero, and for a non multiplying medium the macroscopic fission cross-section, Σ_f is zero. Hence, for a non-absorbing, non-multiplying medium, the buckling is zero from equation 6.9. Now, the time dependent diffusion equation reduces to

$$D\nabla^2\phi = \frac{1}{\upsilon}\frac{\partial\phi}{\partial t} \tag{6.10}$$

The diffusion equation applied to the finite cylindrical core under consideration is

$$\frac{\partial^2\phi}{\partial r^2} + \frac{1}{r}\frac{\partial\phi}{\partial r} + \frac{\partial^2\phi}{\partial z^2} = \frac{1}{D\upsilon}\frac{\partial\phi}{\partial t} \tag{6.11}$$

and the initial and boundary conditions are:

b.c. 1. $\phi(o,z,t) < \infty$, flux is finite on the center line of the cylindrical core.

2. $\phi(R,z,t) = 0$

3. $\phi(r,0,t) = 0$

4. $\phi(r,H,t) = 0$

i.c.: 5. $\phi(r,z,0) = \phi^0$

Solution of a two-dimensional transient problem can be achieved by reducing it into two one-dimensional problems.[51] In this case, the problem is divided into two transient problems in the radial and axial directions. The solution of the equation is:

$$\phi(r,z,t) = 2\phi^0 \cdot \sum_{n=1}^{\infty} \frac{J_0(\lambda_n r)}{\lambda_n R J_1(\lambda_n R)} \exp(-\lambda_n^2 \upsilon D t) \cdot 4 \sum_{m=1}^{\infty} \frac{1}{\lambda_m H} \sin(-\lambda_m^2 z \upsilon D t) \quad (6.12)$$

where $\lambda_n R$ are the roots of $J_0(\lambda_n r)$=0, and $\lambda_m H = (2m-1)$.

6.3 Steady-State, Infinite Cylindrical Reactor with Reflector

The neutron economy in a reactor can be improved by providing a reflector that surrounds the reactor core and reflects neutrons back into the core to enhance fission. Neutrons that pass through the reflector are lost beyond the reactor boundaries and do not return to the reactor core. The addition of a reflector to a reactor decreases the critical size and mass of fuel required within the core. The wall of the reactor core in commercial reactors is composed mild steel over 30 cm thick. The container wall often serves as a neutron reflector.

As an example, a steady-state cylindrical reactor consisting of core radius r_g, and reflector thickness (including extrapolated distance), T, is shown in figure 6.3.

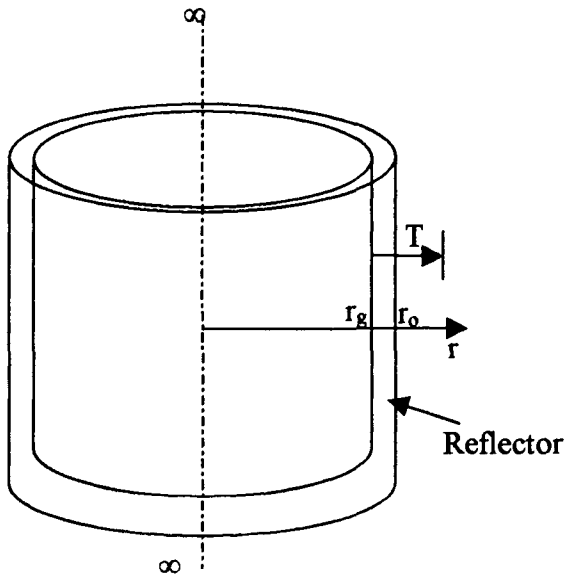


Figure 6.3 Infinite Cylindrical Reactor with a Radial Reflector

The steady state neutron transport equation for an infinite cylindrical reactor where there is no axial dependency is

$$\frac{1}{r}\frac{\partial}{\partial r}r\frac{\partial \phi}{\partial r}+B^2\phi=0 \tag{6.13}$$

and the boundary conditions are

b.c.: 1. $\phi(o)<\infty$, flux is finite on the center line of the cylindrical core.

2. $\phi_r(r_g+T)=0$,

3. $\phi_c(r_g)=\phi_r(r_g)$,

4. $J_c(r_g)=J_r(r_g) \quad or -D_c\frac{\partial \phi}{\partial r} at\ r_g=-D_r\frac{\partial \phi}{\partial r} at\ r_g$

where ϕ_c, ϕ_r are the neutron flux in the core and reflector, respectively, and J_c, J_r are the neutron current densities.

The neutron transport equation is applied separately to the core and in the reflector. Boundary conditions 3 and 4 provide that the neutron flux and current density calculated at the core/reflector interface must agree. The neutron transport equation applied to the core is

$$\frac{\partial^2 \phi_c}{\partial r^2} + \frac{1}{r}\frac{\partial \phi_c}{\partial r} + B_c^{\,2}\,\phi_c = 0 \tag{6.14}$$

where B_c is the buckling in the core. The solutions to this differential equation are in the form of Bessel functions of the first kind of order zero. Therefore, the general solution of the equation is

$$\phi_c = C_1 J_0(B_c r) + C_2 Y_0(B_c r) \tag{6.15}$$

where C_1, C_2 are constants. Applying B.C 1 gives the solution of the equation 6.15 for the neutron flux variation in the core in radial direction as

$$\phi_c = C_1 J_0(B_c r) = \phi_{\max} J_0(B_c r) \tag{6.16}$$

where $\phi_{\max}$ is the maximum flux in the reactor core.

In reflectors, there is no neutron multiplying medium ($k_\infty = 0$), hence B_r, the buckling in the reflector becomes

$$B_r = \frac{k_\infty - 1}{L_r^2} = -\frac{1}{L_r^2} \tag{6.17}$$

where L_r is the diffusion length in the reflector. Now, the neutron transport equation in the reflector is

$$\frac{\partial^2 \phi_r}{\partial r^2} + \frac{1}{r}\frac{\partial \phi_r}{\partial r} - \frac{1}{L_r}\phi_r = 0 \tag{6.18}$$

Solutions to this equation are in the form of modified Bessel functions of order zero. Therefore the general solution of equation 6.18 is:

$$\phi_r = C_3 I_0\left(\frac{r}{L_r}\right) + C_4 K_0\left(\frac{r}{L_r}\right) \tag{6.19}$$

where C_3, C_4 are constants. Applying boundary conditions 2, 3, and 4 and taking the first root of the Bessel's function from the eigen condition yields the solution of equation 6.19 for the neutron flux distribution in the core and in the reflector in radial direction as

$$\phi_c = \phi_{\max} J_0\left(\frac{2.40r}{r_o}\right) \tag{6.20}$$

$$\phi_r = \frac{\phi_{\max} J_0\left(\frac{2.405r_g}{r_o}\right)}{I_0\left(\frac{r_g}{L_r}\right) - \frac{I_0\left(\frac{r_g+T}{L_r}\right)}{K_0\left(\frac{r_g+T}{L_r}\right)} K_0\left(\frac{r_g}{L_r}\right)} \left[I_0\left(\frac{r}{L_r}\right) - \frac{I_0\left(\frac{r_g+T}{L_r}\right)}{K_0\left(\frac{r_g+T}{L_r}\right)} K_0\left(\frac{r}{L_r}\right) \right] \tag{6.21}$$

6.4 Variation in Xenon Concentration within the Reactor Over Time

As explained in Chapter 3, some fission products and their decay products have very large absorption cross-sections and generation of these isotopes can have a significant impact on reactor behavior. Xenon (^{135}Xe), a fission product, is produced as a fission fragment and also from the decay of the other fission products, such as iodine. During reactor operation, xenon (^{135}Xe) continues to build up until it reaches a steady-state concentration, where the generation and the loss from neutron absorption and self decay

are balanced. In this section, variation in the concentration of xenon over time is analyzed. The variation of xenon over time is given by the following equations

$$\frac{dI}{dt} = \gamma_I \overline{\Sigma}_f \phi_T - \lambda_I I \tag{6.22}$$

$$\frac{dX}{dt} = \lambda_I I + \gamma_X \overline{\Sigma}_f \phi_T - \lambda_X X - \overline{\sigma}_{aX} \phi_T X \tag{6.23}$$

Detailed discussion about the production of these isotopes, along with the notation used, was described in Chapter 3. By integrating equations 6.22 and 6.23 with respect to time and by noting that $\phi_T = 0$ after shutdown, the variation of iodine and xenon concentrations after the reactor is shut down is given by:

$$I(t) = I_0 e^{-\lambda_I t} \tag{6.24}$$

$$X(t) = X_0 e^{-\lambda_X t} + \frac{\lambda_I I_0}{\lambda_I - \lambda_X}(e^{-\lambda_X t} - e^{-\lambda_I t}) \tag{6.25}$$

where I_0, X_0 are the concentrations of iodine and xenon, respectively. The fission product yield from thermal fission for different isotopes and decay constants of the fission product poisons described in the above equations were given in Tables 3.1 and 3.2.

6.5 Code Validation

The analytical solutions derived in this chapter were employed to validate the steady-state and transient behavior of a reactor as predicted by ADSTRANS. The next chapter shows the comparison of analytical solutions to code predictions.

CHAPTER 7

RESULTS

In this chapter, "ADSTRANS", "MCNPX" simulations, analytical and experimental results are presented. The "ADSTRANS" simulations were validated by comparing them with a series of analytical solutions for both transient and steady-state reactor behavior. The "MCNPX" simulations were validated by comparing them with experimental results. Experiments were conducted at two laboratories to provide validation that MCNPX adequately predicts neutron production based on electron or proton accelerators.

7.1 Comparison of MCNPX Predictions with Experimental Results

The MCNPX predictions of photo-neutron production were compared with experimental results performed at the Idaho Accelerator Center (IAC) using an electron accelerator. This work was done as part the requirements for the author's M.S. degree in 2002. The work reported in this dissertation was a continuation of research initiated in the M.S. project.

A lead target with 5.08 cm diameter and 127 cm depth has been irradiated with an 18 MeV electron beam from a 20 MeV electron linear accelerator (LINAC). Neutron production in the target was measured through lead activation foils located at different

depths within the target. The arrangement of the target and the accelerator are as shown in the figure 7.1.

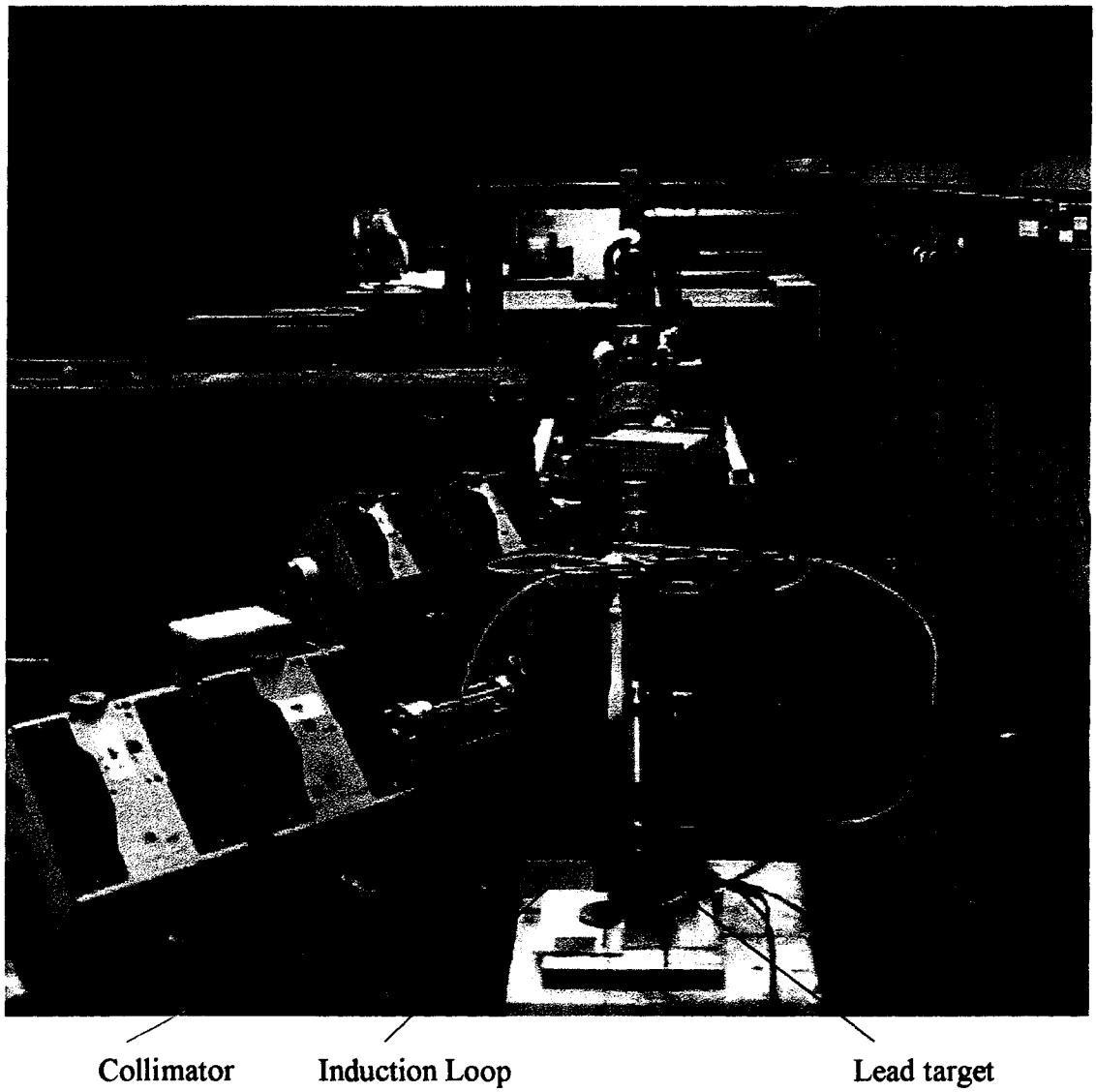


Figure 7.1 Electron Beam from 20 MeV LINAC Impacting a Lead Target

MCNPX version 4.2j was used to simulate the photo-neutron production in the lead target from the mono-energetic electron beam of 18 MeV. In the experiments, steering magnets and a carbon collimator were used to produce the mono-energetic electron beam. The

LINAC produced an electron beam with a repetition rate of 60 Hz and a pulse width of 2 microseconds. The experimental neutron production was compared to MCNPX simulations as shown in table 7.1 and in figure 7.2. The neutron production per incident

Table 7.1 MCNPX and Experimental Comparison for Photo-Neutron Production.

Foil No.	Foil Position (cm)	Simulation Results (neutrons/electron-cm)	Experimental Results (neutrons/electron-cm)	% deviation
1	0.69	9.33×10^{-4}	9.43×10^{-4}	-1.1
2	1.45	5.70×10^{-4}	6.30×10^{-4}	-9.5
3	2.15	3.09×10^{-4}	4.13×10^{-4}	-25.2
4	2.84	1.94×10^{-4}	2.40×10^{-4}	-19.2

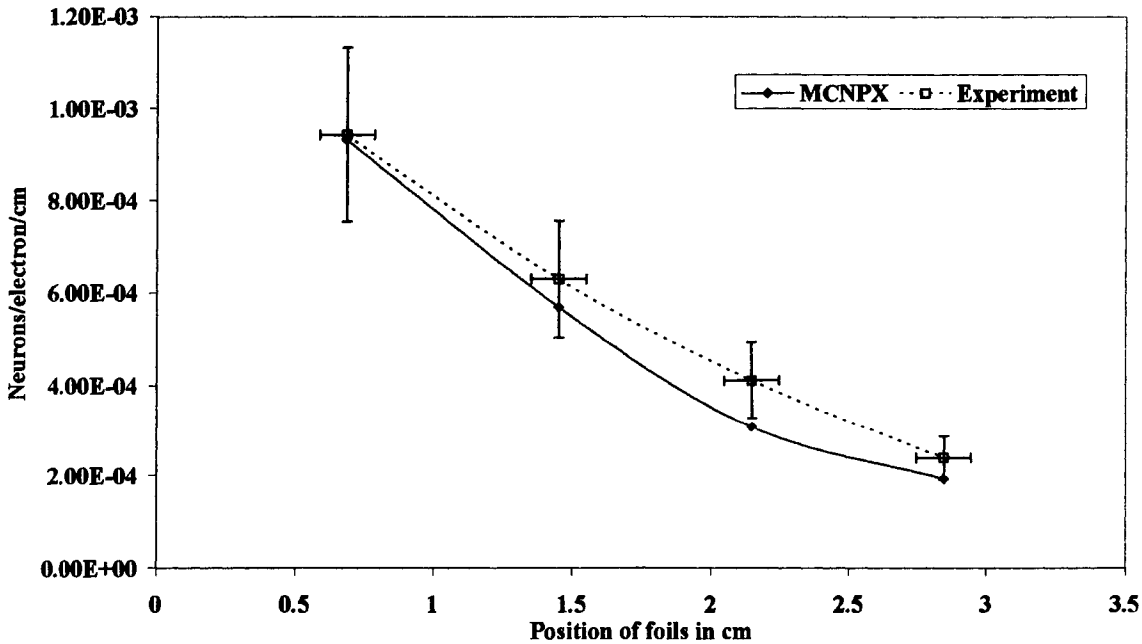


Figure 7.2 Comparison between MCNPX and Experimental Results

electron per centimeter thickness of target as predicted by MCNPX agreed with the experimental results roughly within the error bars. The total neutron production was found by integrating the curves in figure 7.2. The total number of neutrons produced per incident electron within the first 3 cm of the target depth from the experiments was 1.399×10^{-3}, while MCNPX simulations predicted a value of 1.246×10^{-3}. The total neutron production in the lead target predicted by the MCNPX simulations agreed within 10.9% of the experimental results.

One of our group members has done experiments at Los Alamos national laboratory to compare spallation neutron production. Neutron yield from a 20 cm diameter, 50 cm long lead-bismuth target irradiated by 800 MeV protons were compared with MCNPX predictions. It was reported that the predictions by MCNPX were in good agreement with experimental results.

7.2 Validation of Simulation Results through Comparison with Analytical Results

The simulations from 'ADSTRANS' were compared against the analytical solutions. The code was verified in different scenarios such as steady state neutron flux distribution, time dependent neutron flux variation in finite and infinite reactors with bare and reflected boundaries. The fission poisons variation over time predicted by the code was also verified against the analytical solution.

7.2.1 Spatial Neutron Flux Distribution in a Steady-State, Finite, Bare Cylindrical Reactor

A steady-state cylindrical core reactor without reflector was considered with the details of the geometry given in figure 6.1. The neutron flux distribution in the radial and

axial direction of the reactor was obtained from simulation code and analytical solutions. The general simulation code was for a finite cylinder core with neutron reflectors around the core. In this specific case the reflectors are removed and the entire geometry of the reactor was treated as core. The analytical solution of the neutron flux distribution was

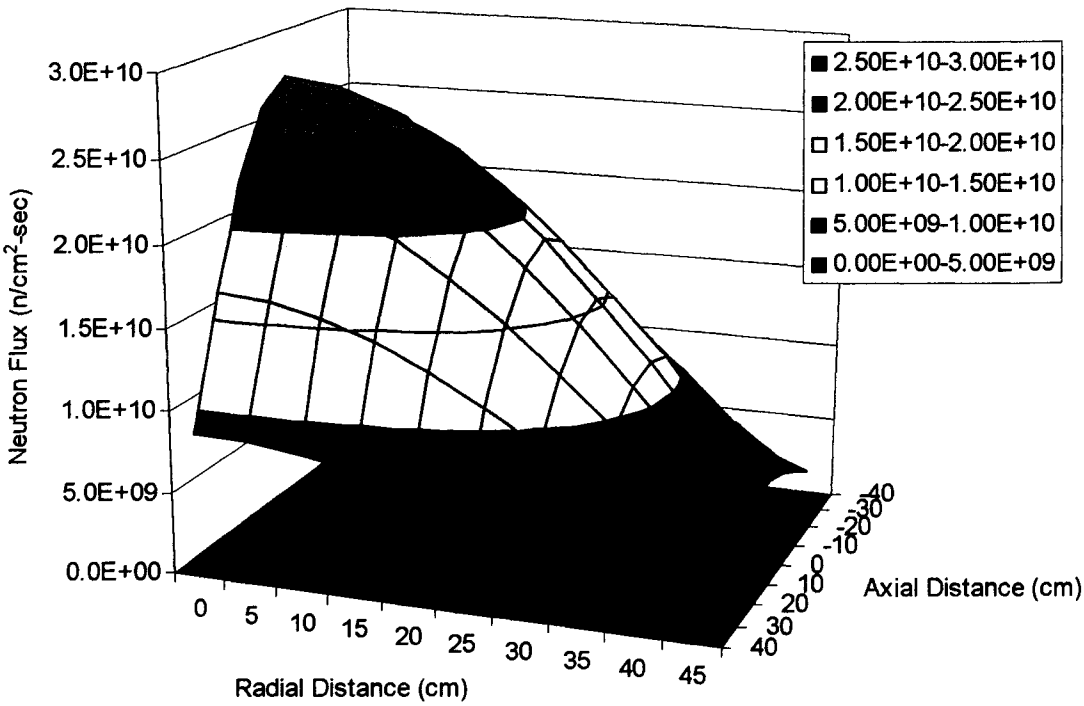


Figure 7.3 Neutron Flux Distribution from the Analytical Solution

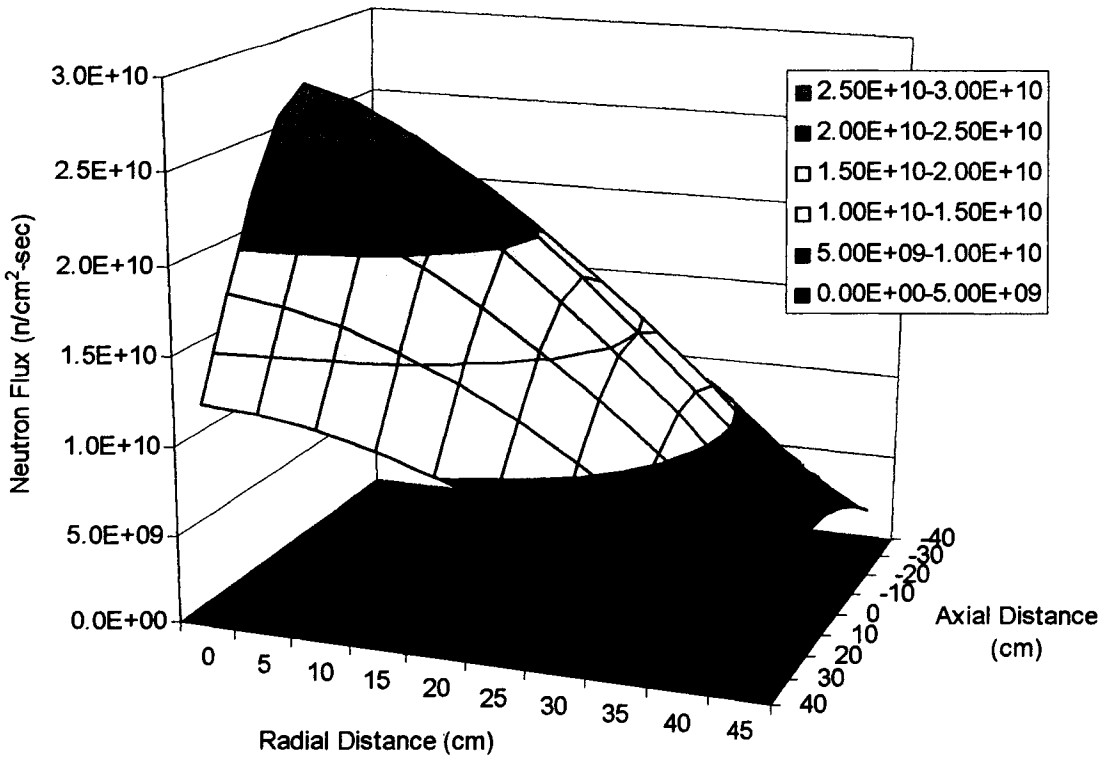


Figure 7.4 Neutron Flux Distributions from the Numerical Computer Code

obtained solving the diffusion equation as presented in Chapter 6. The neutron flux distribution in the reactor from simulation and analytical solution are presented in the figures 7.3 and 7.4.

7.2.2 Transient Neutron Flux in a Finite Bare Cylindrical Reactor

A finite bare cylindrical core reactor with the geometry boundary conditions shown in figure 6.2 (Chapter 6) was considered, and to simplify the analytical solution non-absorbing, non-multiplying material was considered for the reactor. The neutron flux variation at a particular location of the reactor over time from simulation was compared with the analytical solution. The simulation code was modified to this specific case by removing the neutron reflectors, by forcing the boundaries to have a constant neutron flux as specified in the problem through the reactor operation and assigning a uniform neutron

flux throughout the reactor. In this particular problem, the neutron absorption and fission cross-sections are set to zeros to make it as a simple diffusion problem. The analytical solutions were obtained as given in Chapter 6 for the time-dependent neutron flux variation at a particular location. The neutron flux variation over time (up to 60 time steps of simulation) from simulation and analytical solution are compared as shown in the figure 7.5. The simulation results were in good agreement with the analytical results, within 10%, as shown in the table 7.2.

Table 7.2 Comparison of Time Dependent Neutron Flux

Time (sec)	Analytical Solution Φ (n/cm^2-s)	Numerical Solution Φ (n/cm^2-s)	% Error
0	1.00E+13	1.00E+13	0.0
0.0001	9.87E+12	9.25E+12	6.3
0.0002	8.69E+12	7.9E+12	9.1
0.0004	5.22E+12	4.93E+12	5.6
0.0006	2.88E+12	2.8E+12	2.7
0.0008	1.56E+12	1.54E+12	1.5
0.001	8.47E+11	8.37E+11	1.2
0.0015	1.83E+11	1.81E+11	1.2
0.002	3.94E+10	3.89E+10	1.4
0.0025	8.52E+09	8.37E+09	1.7
0.003	1.84E+09	1.80E+09	1.9
0.0035	3.97E+08	3.88E+08	2.1
0.004	8.56E+07	8.36E+07	2.3
0.0045	1.85E+07	1.80E+07	2.6
0.005	3.99E+06	3.88E+07	2.8

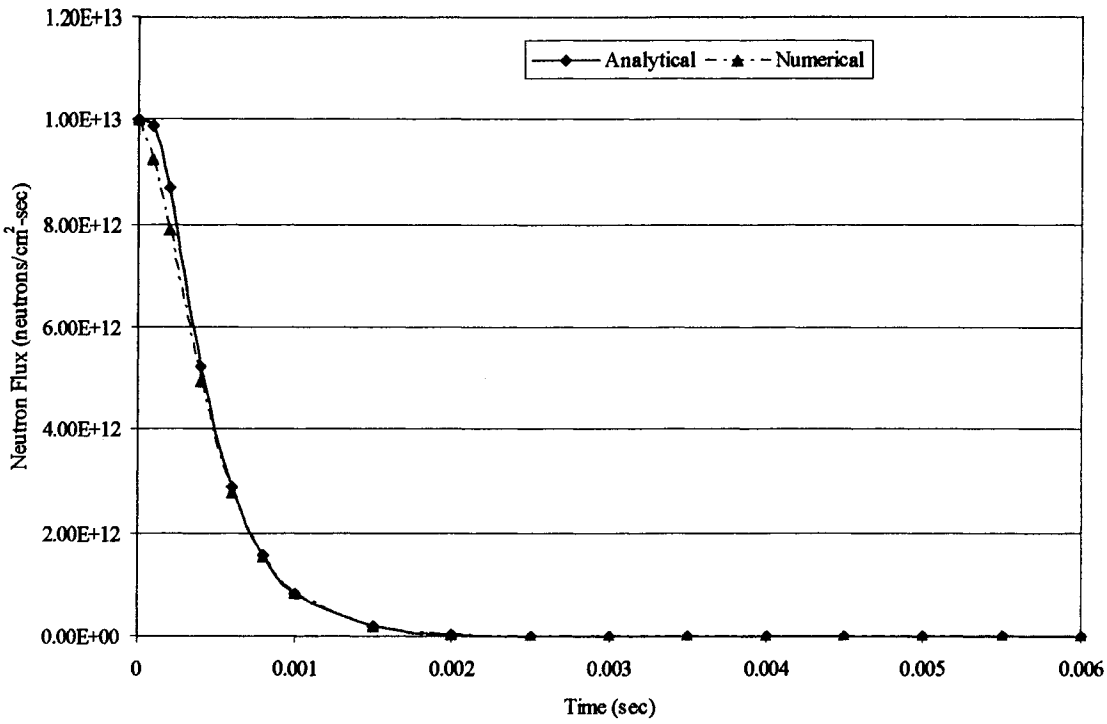


Figure 7.5 Comparison of Transient Neutron from Analytical and Numerical Results

7.2.3 Neutron Flux Distribution in a Steady State, Infinite Cylindrical Reactor with Reflector

A steady state infinite cylindrical reactor with reflectors was considered as shown in the figure 6.3 (Chapter 6). The geometry and boundary conditions of the reactor are given in Chapter 6. The neutron flux variation in the radial direction predicted by the simulation was compared with that of the analytical solutions (neutron flux was constant in the axial direction for an infinite cylinder). In the code, the length of the reactor was set very long compared to the radius of the reactor, by which infinite length behavior of the reactor was achieved. The neutron fluxes predicted by the code at different radial distances in the reactor were in good agreement with the analytical solutions, within 5% deviation, as shown in the figure 7.6 and in the table 7.3.

Table 7.3 Comparison between Analytical Numerical Results

Radial Distance from the Center (cm)	Analytical Φ (n/cm^2-sec)	Numerical Φ (n/cm^2-sec)	% Error
0	2.574×10^{10}	2.574×10^{10}	0
5	2.537×10^{10}	2.498×10^{10}	-1.5
10	2.427×10^{10}	2.355×10^{10}	-3.0
15	2.250×10^{10}	2.161×10^{10}	-3.9
20	2.012×10^{10}	1.927×10^{10}	-4.2
25	1.724×10^{10}	1.659×10^{10}	-3.8
30	1.399×10^{10}	1.363×10^{10}	-2.6
35	1.049×10^{10}	1.035×10^{10}	-1.3
40	6.898×10^{9}	6.659×10^{9}	-3.5
45	1.310×10^{9}	1.286×10^{9}	-1.9

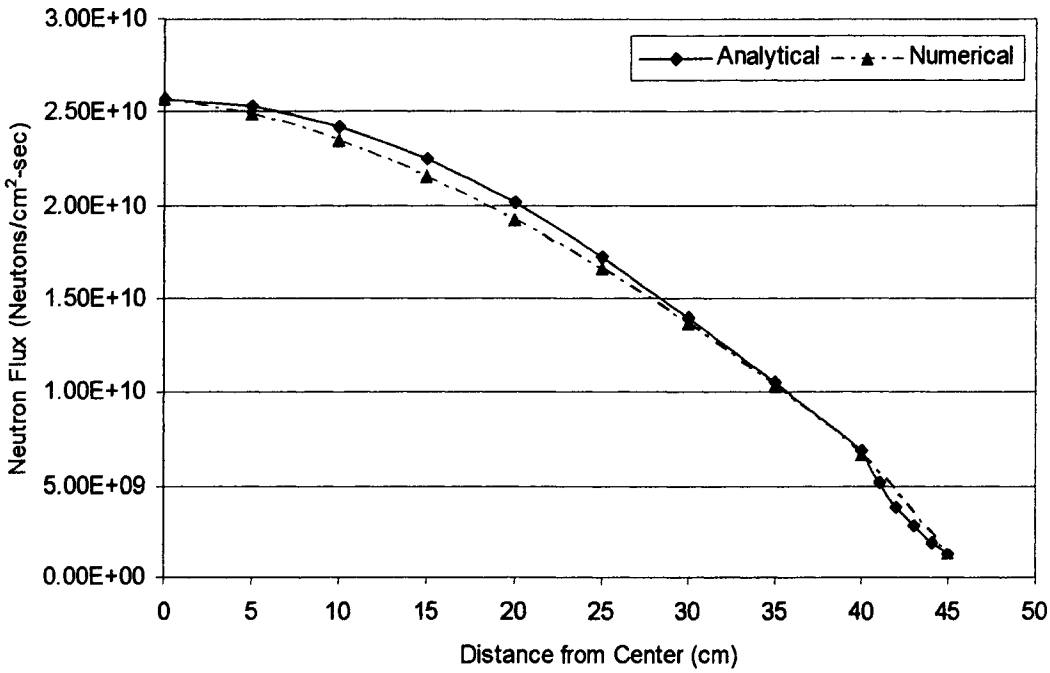


Figure 7.6 Comparison between Numerical and Analytical Results

7.2.4 Neutron Poison Concentration after Reactor Shutdown

The variation of Xenon (^{135}Xe), a fission product neutron poison, concentration changes in the reactor even after shutdown. This was due to the decay of other fission products to xenon and self decay of xenon. In this section the variation of xenon over time after the reactor shutdown predicted by the code was compared with the analytical results. In the simulations, the reactor was shutdown after a half-an-hour operation, and the xenon concentration was observed.

The ^{135}Xe concentration predicted by the code was in good agreement with the analytical results, within 3.6 %, as shown in the table 7.4. and figure 7.7.

Table 7.4 Comparison between ^{135}Xe Number Densities from Numerical and Analytical Results, after Reactor Shutdown at t = 0

Time (sec)	Analytical (n/cm^3)	Numerical (n/cm^3)	% Error
0	2.09E+19	2.09E+19	0
200	2.25E+19	2.28E+19	-1.2
400	2.40E+19	2.45E+19	-2.0
600	2.55E+19	2.62E+19	-2.6
800	2.70E+19	2.79E+19	-3.0
1000	2.85E+19	2.95E+19	-3.3
1200	3.00E+19	3.10E+19	-3.4
1400	3.14E+19	3.25E+19	-3.6
1600	3.28E+19	3.40E+19	-3.4
1800	3.42E+19	3.53E+19	-3.3

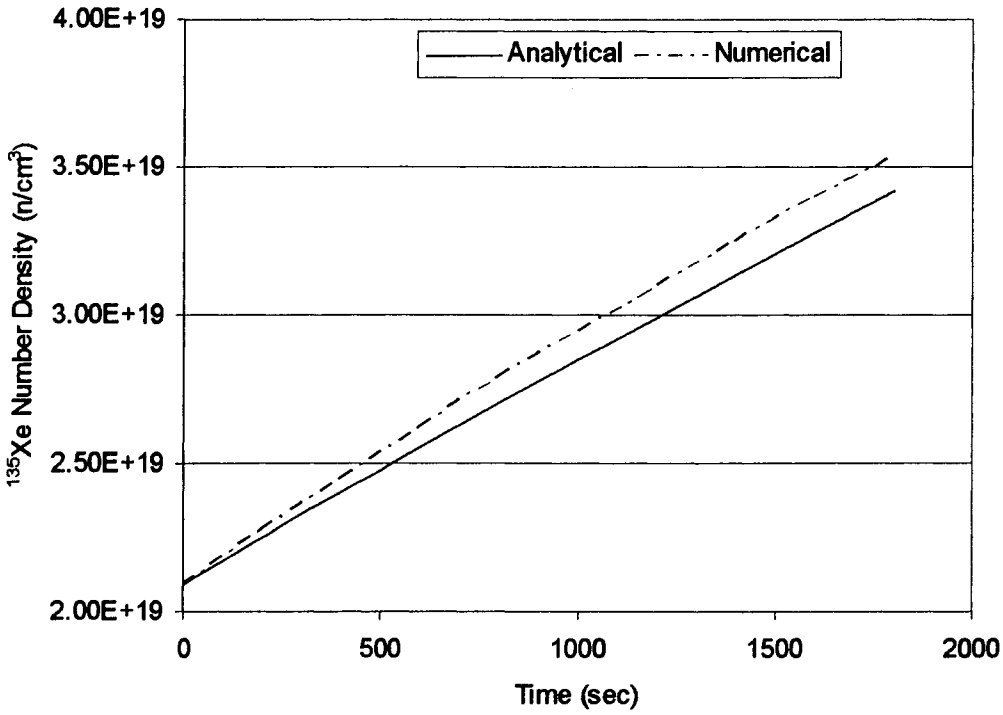


Figure 7.7 Comparison between ^{135}Xe Number Densities from Numerical and Analytical Results, after Reactor Shutdown at t = 0

7.3 Numerical Comparisons

Two important variables of a numerical difference code which can affect the stability of the simulations are cell density and time step. In this section, stability of 'ADSTRANS' simulations are compared, for different cell densities and for different time steps. The neutron flux distribution and transient behavior are compared for cell densities of 10 by 10 and 20 by 20; for time steps of 0.1 milliseconds and 0.05 milliseconds.

7.3.1 Cell Densities (10 by 10 and 20 by 20 mesh)

The neutron flux distribution in the accelerator driven system (ADS) and the time dependent neutron flux variation in the ADS, predicted by code for 10 by 10 and 20 by 20 cell densities, were compared. The neutron flux distribution in the reactor predicted by the code through different cell densities were in good agreement, within 12 %, as shown in figures 7.8, 7.9, 7.10 and table 7.5. The time dependent neutron fluxes predicted by the code through different cell densities were also in very good agreement as shown in figures 7.11, 7.12 and table 7.6.

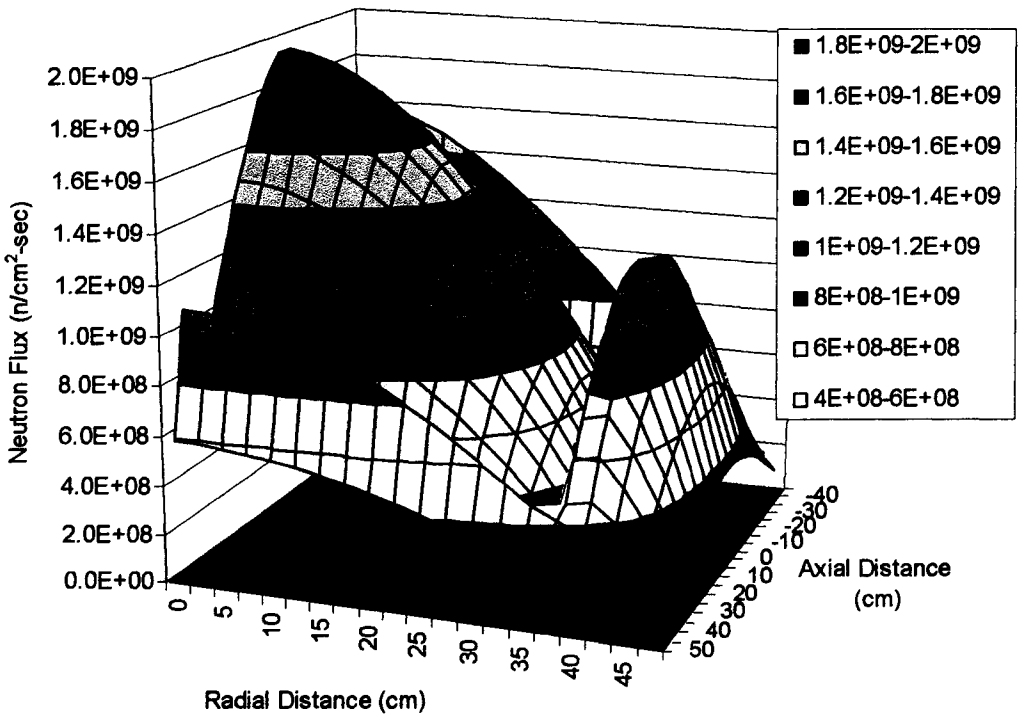


Figure 7.8 Neutron Flux Distribution from 20 by 20 Cell Density

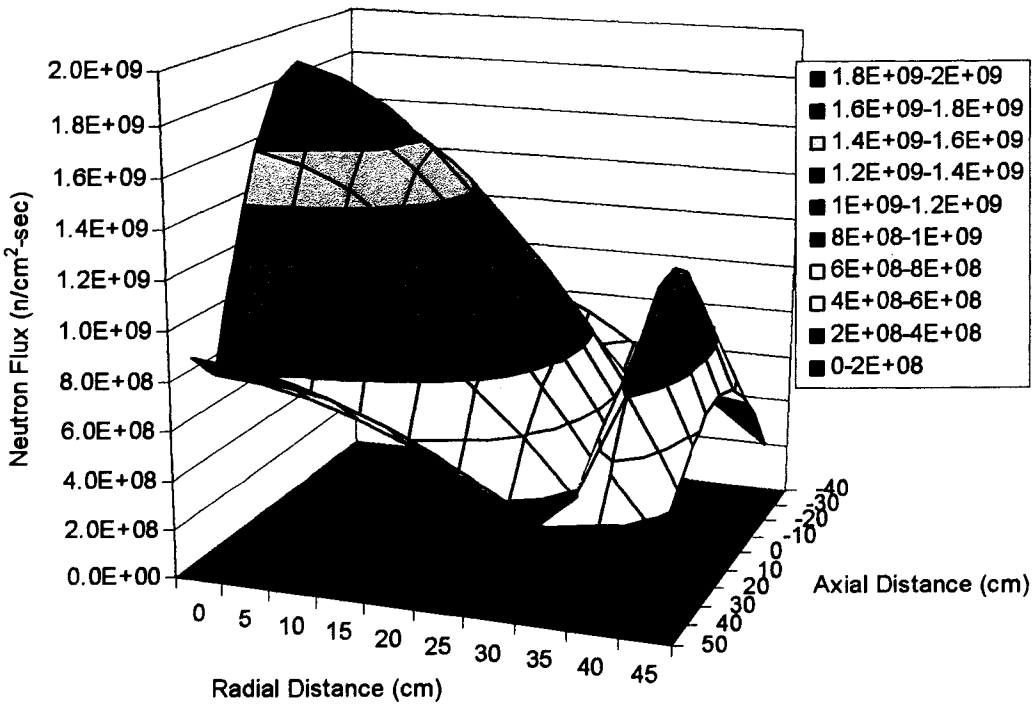


Figure 7.9 Neutron Flux Distribution from 10 by 10 Cell Density

Table 7.5 Comparison of Neutron Flux between 10 by 10 and 20 by 20 Cell Densities

Radial Distance (cm)	Cell Density 10 by 10 Φ (n/cm^2-sec)	Cell Density 20 by 20 Φ (n/cm^2-sec)	% Deviation
0	1.89E+09	1.89E+09	0.21
5	1.84E+09	1.84E+09	-0.41
10	1.72E+09	1.74E+09	-0.80
15	1.57E+09	1.58E+09	-0.88
20	1.37E+09	1.38E+09	-0.49
25	1.13E+09	1.13E+09	0.70
30	8.61E+08	8.29E+08	3.74
35	5.54E+08	4.90E+08	11.62
40	1.16E+09	1.17E+09	-1.41
45	6.53E+08	6.66E+08	-1.92

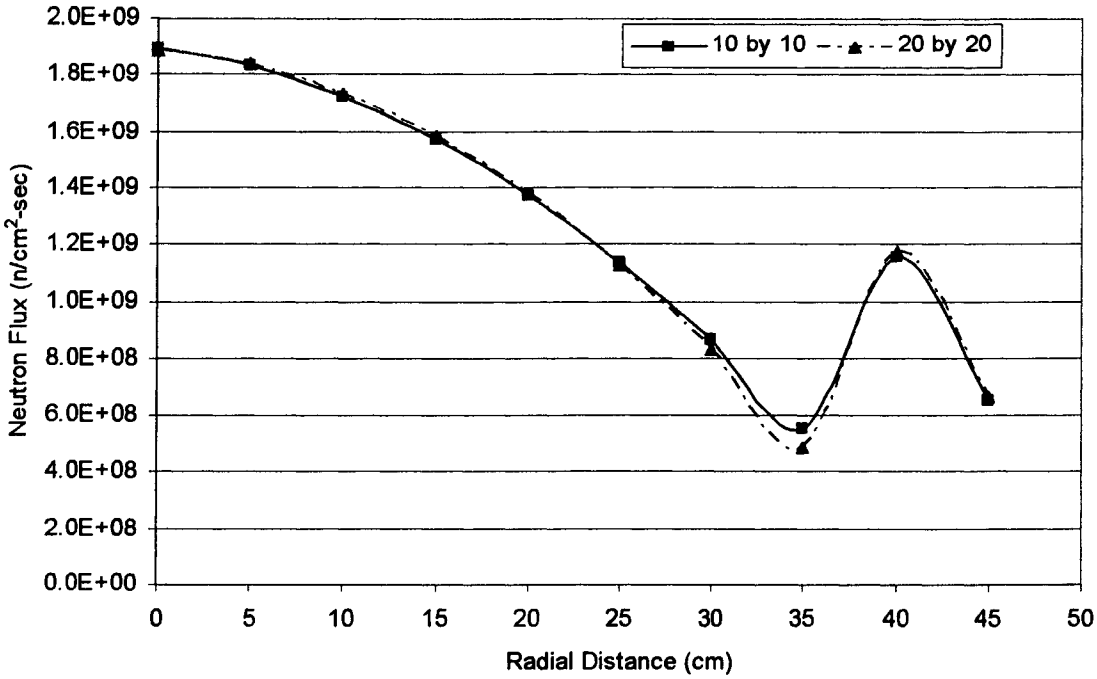


Figure 7.10 Neutron Flux Variation in the Radial Direction from Different Cell Densities

Table 7.6 Comparison of Neutron Flux Variation over Time between the 20 by 20 and 10 by 10 Cell Densities

Time (sec)	20 by 20 Φ (n/cm^2-sec)	10 by 10 Φ (n/cm^2-sec)	% Deviation
0	2.63E+10	2.36E+10	10.5
0.0001	1.90E+09	1.78E+09	6.2
0.0002	5.68E+07	5.50E+07	3.1
0.0003	9.81E+06	9.71E+06	1.0
0.0004	8.83E+06	8.73E+06	1.1
0.0005	8.81E+06	8.71E+06	1.1
0.0006	8.81E+06	8.71E+06	1.1
0.0007	8.81E+06	8.71E+06	1.1
0.0008	8.81E+06	8.71E+06	1.1
0.0009	8.81E+06	8.71E+06	1.1
0.001	8.81E+06	8.71E+06	1.1
0.0011	8.81E+06	8.72E+06	1.1
0.0012	8.81E+06	8.72E+06	1.1
0.0013	8.81E+06	8.72E+06	1.1
0.0014	8.82E+06	8.72E+06	1.1
0.0015	8.82E+06	8.72E+06	1.1
0.0016	8.82E+06	8.72E+06	1.1

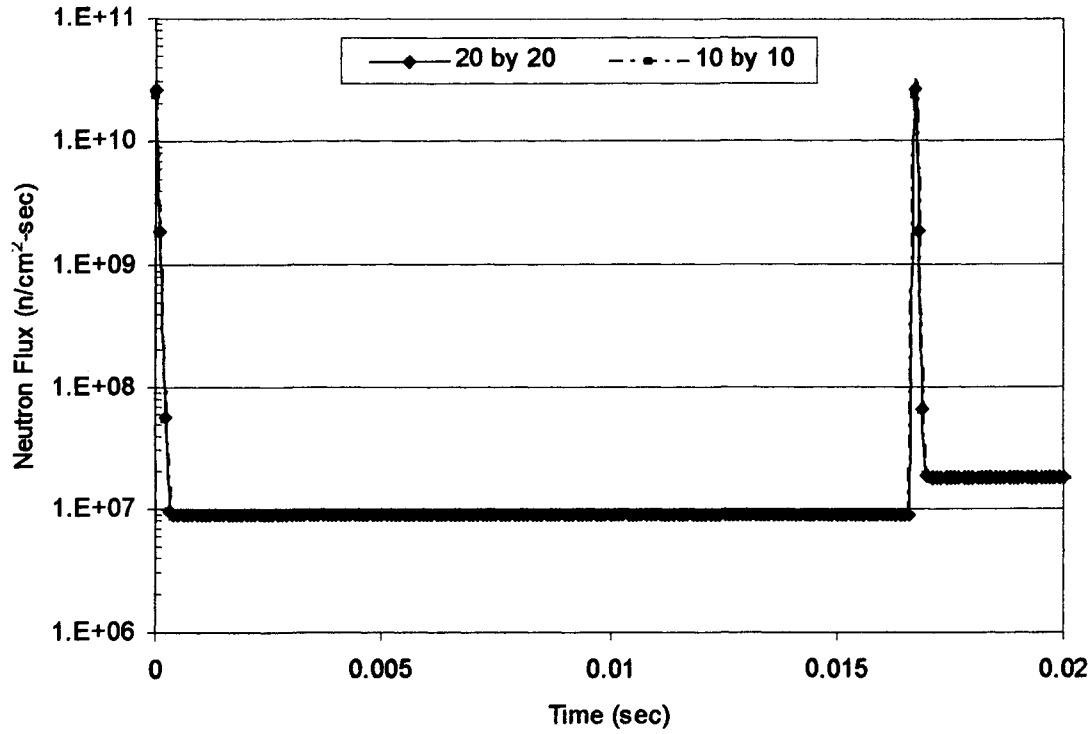


Figure 7.11 Neutron Flux in the Reactor over Time Based On Different Cell Densities

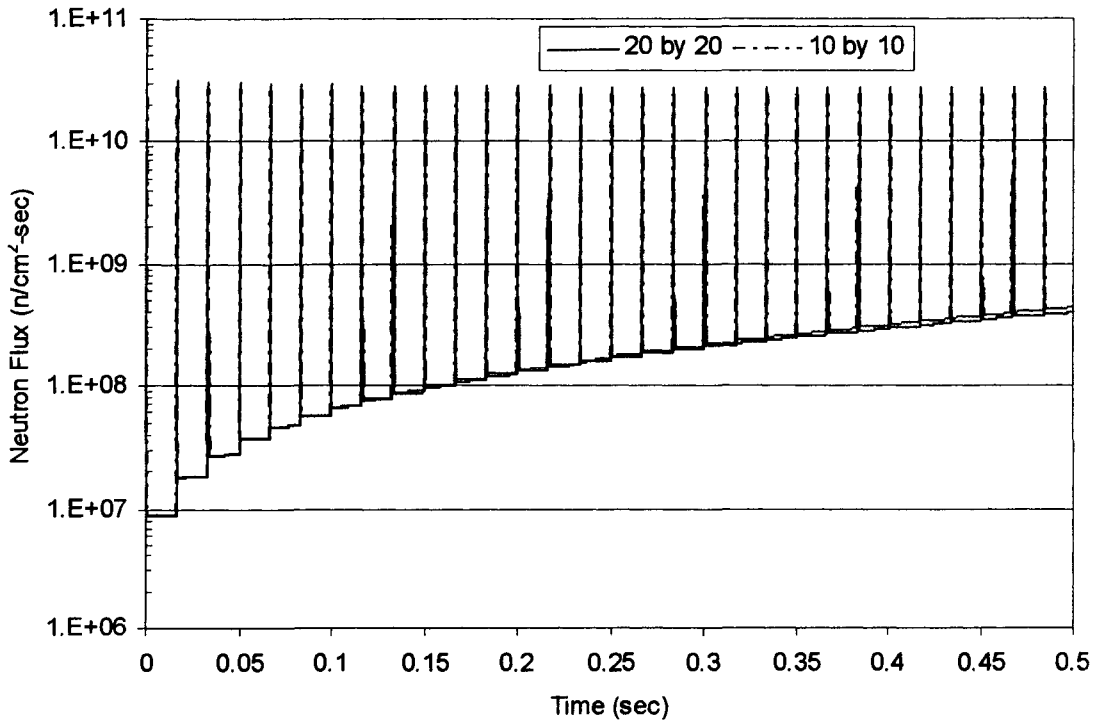


Figure 7.12 Neutron Flux in the Reactor over 0.5 sec Based On Different Cell Densities

7.3.2 Neutron Flux Dependency upon Timestep (0.1 milliseconds and 0.05 milliseconds)

The neutron flux distribution in the ADS and the time dependent neutron flux variation in the ADS, predicted by the code for time step 0.1 milliseconds and 0.05 milliseconds were compared. The neutron flux distributions predicted by the code through the two different time steps are in good agreement, within 10% deviation, as shown in table (Appendix III) and figures 7.13 and 7.14. The time dependent neutron flux predicted by the code through the two different time steps were also in good agreement except for a very short time (four time steps) in the beginning as shown in table. 7.7. and figures 7.15 and 7.16.

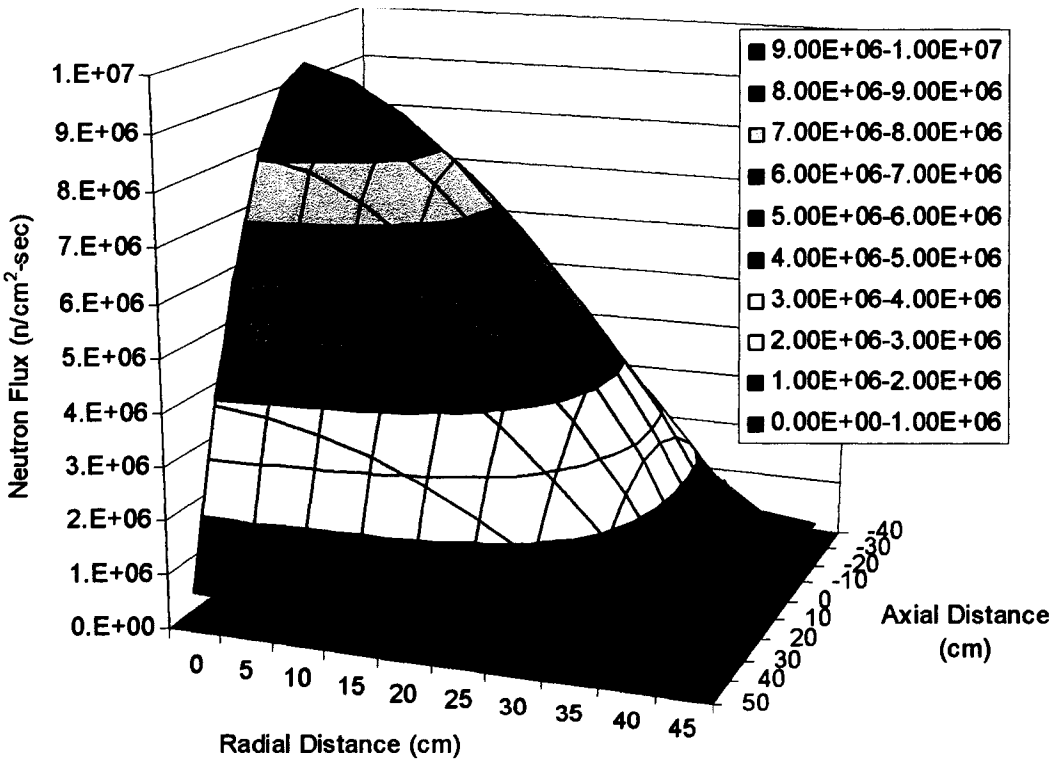


Figure 7.13 Neutron Flux Distribution in the Reactor for 0.05 msec Timestep

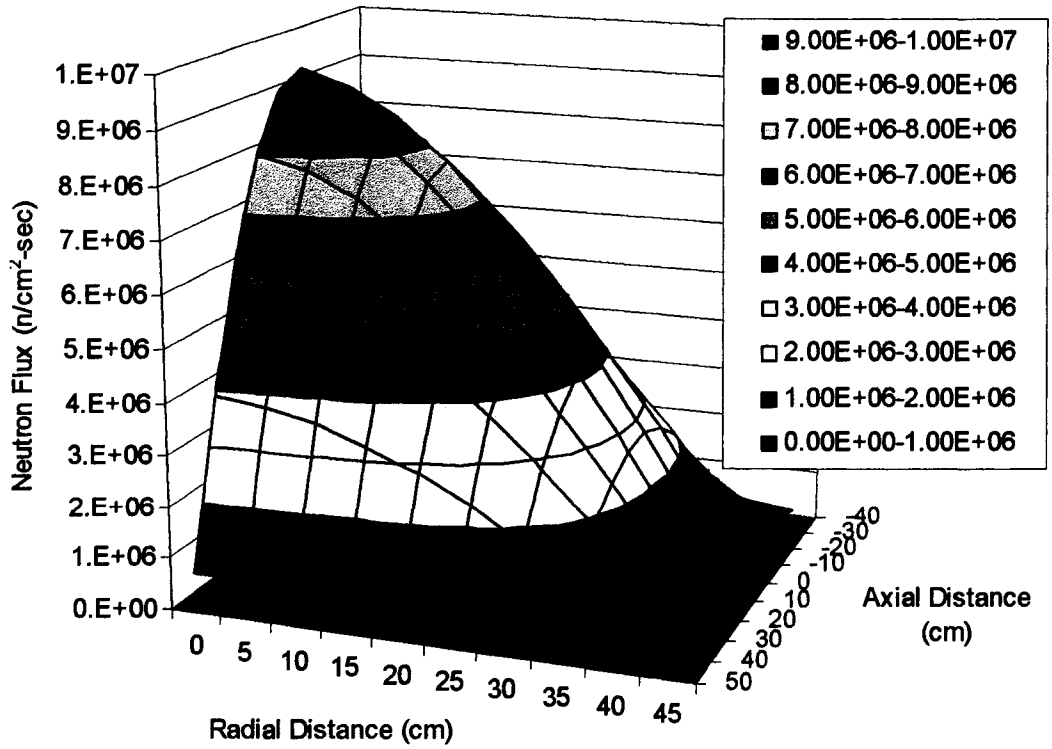


Figure 7.14 Neutron Flux Distribution in the Reactor for 0.1 msec Timestep

Table 7.7 Comparison of Time-Dependent Neutron Flux, 0.1 and 0.05 msec Timestep

Time (sec)	Timestep 0.1 msec Φ (n/cm^2-sec)	Timestep 0.05 msec Φ (n/cm^2-sec)	% Deviation
0	0.00E+00	0.00E+00	0
0.0001	1.15E+10	1.33E+09	88.4
0.0002	5.33E+08	3.00E+06	99.4
0.0003	1.17E+07	1.63E+06	86.1
0.0004	1.79E+06	1.63E+06	9.1
0.0005	1.63E+06	1.63E+06	0.2
0.0006	1.63E+06	1.63E+06	0.0
0.0007	1.63E+06	1.63E+06	0.0
0.0008	1.63E+06	1.63E+06	0.0
0.0009	1.63E+06	1.63E+06	0.0
0.001	1.63E+06	1.63E+06	0.0
0.0011	1.63E+06	1.63E+06	0.0
0.0012	1.63E+06	1.63E+06	0.0
0.0013	1.63E+06	1.63E+06	0.0
0.0014	1.63E+06	1.63E+06	0.0
0.0015	1.63E+06	1.63E+06	0.0
0.0016	1.63E+06	1.63E+06	0.0
0.0017	1.63E+06	1.63E+06	0.0

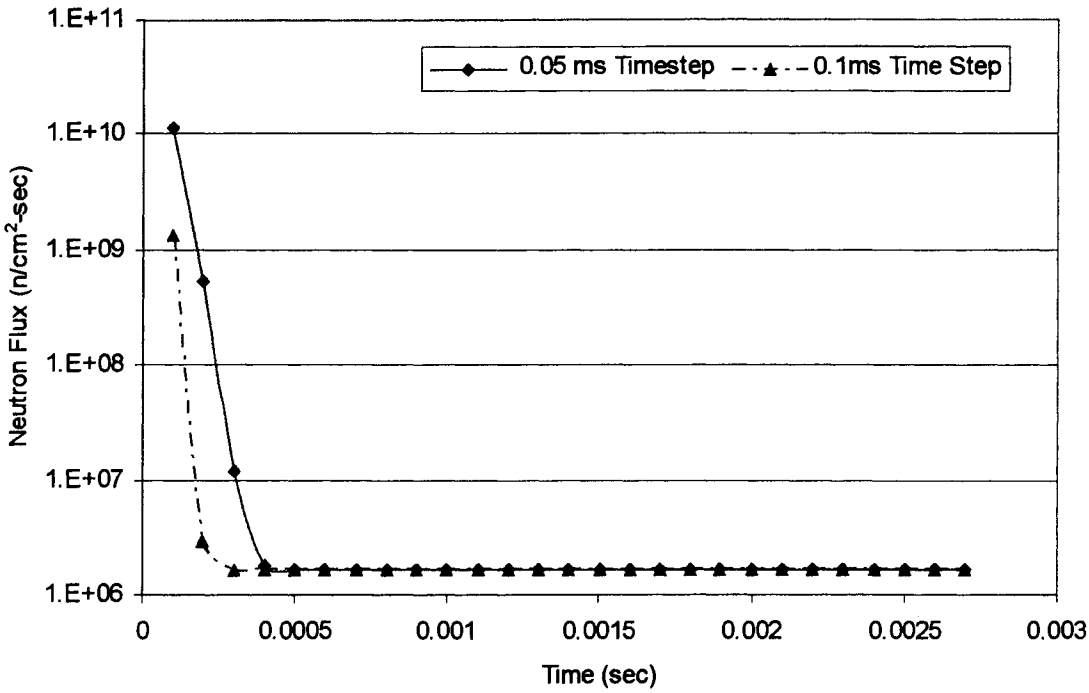


Figure 7.15 Comparison of Neutron Flux Variation over Time for Different Timesteps

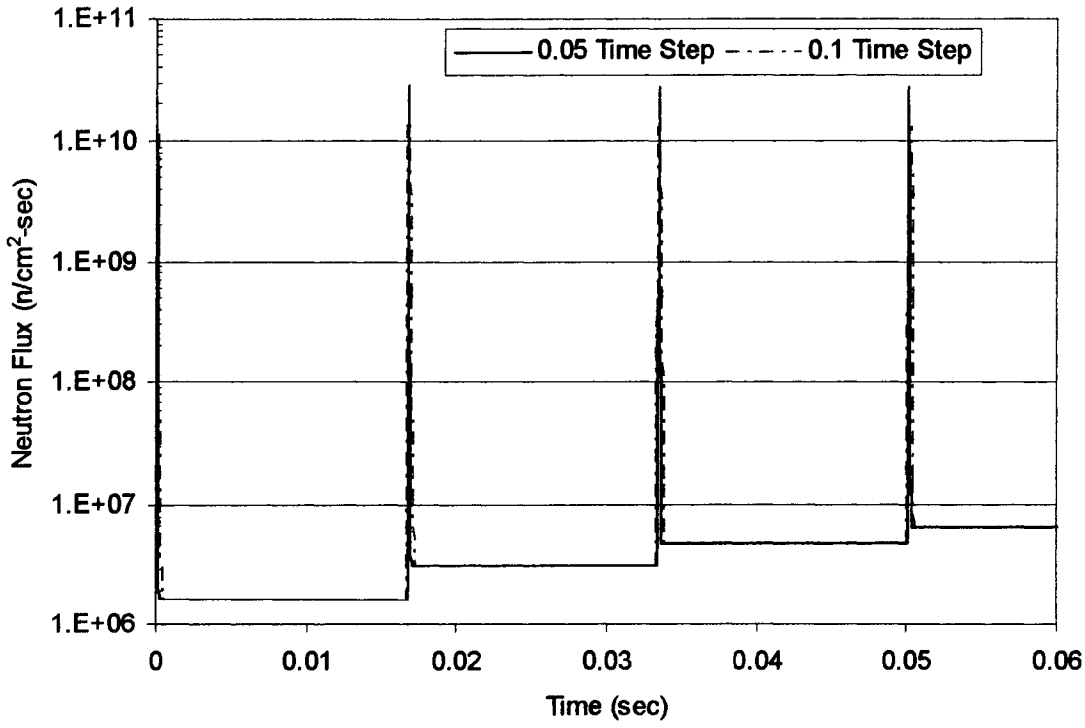


Figure 7.16 Comparison of Neutron Flux Over Time for Different Timesteps

7.4 "ADSTRANS" Simulation Results

The simulation results from "ADSTRANS" for cylindrical reactor coupled to a pulsed proton or electron accelerator are presented. In the simulation described below, the target was made out of lead and was located at the center of the reactor on the axis line. A steel reflector was added around the circumference, top and bottom of the reactor as shown in figure 7.17. At present, there are no accelerator-driven reactors under operation. A test case with the following specifications was considered:

Nuclear Reactor:

Core Specifications:

radius:	40 cm
length:	80 cm
material:	uniform material distribution in each cell (H_2O, U-235, U-238, Pu-249, Np-237, Am-241, Cm-245, the concentration for each case is described below)

Reflector Specifications:

thickness:	10 cm
reflector material:	steel, (Fe=90%, Cr=10%)

Particle Accelerator:

particle:	protons or electrons
energy:	1000 MeV
current:	0.001 mA
pulse charge:	$4.96 \cdot 10^{-8}$ coulombs
pulse width:	$2 \cdot 10^{-6}$ sec
frequency:	60 Hz

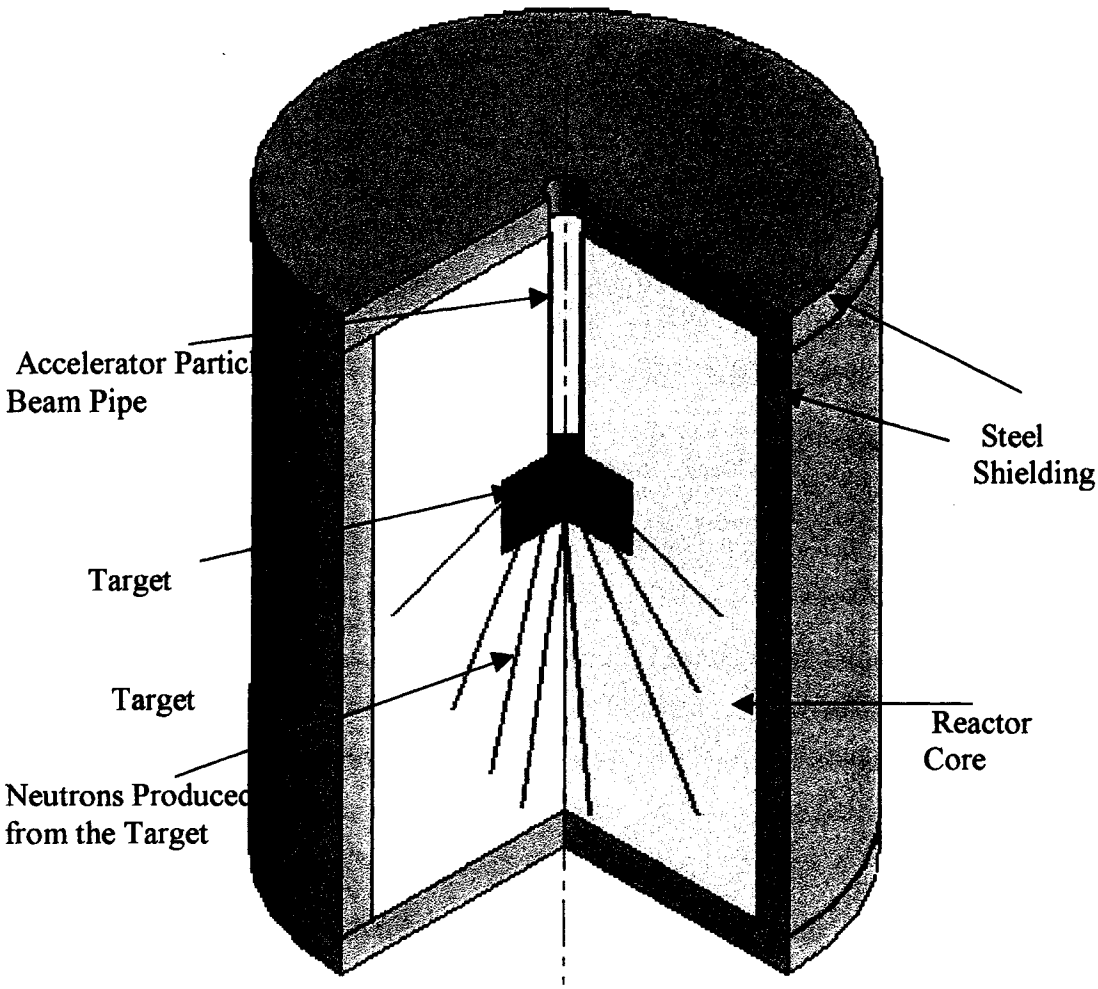


Figure 7.17 Accelerator Driven Nuclear Reactor with Target at the Center

Cross-sections were obtained from ENDF\B libraries using cross-section plotting and printing commands through MCNPX. The cross-sections for the materials used were given in Appendix II. Three energy groups (thermal, epithermal, fast) were considered in these simulations. Energies of these groups were decided based upon the cross-section plots of plutonium-239, which was one of the important materials in transmutation. Other details of the simulation are given in Appendix IV.

Simulations were run on AMD 3200+ Athlon 64-bit machines with a 2.18 GHz speed processor, 2 GB RAM and the LINUX SUSE 9.3 operating system. On these machines,

a general case with 10 by 10 cell density and 0.1 millisecond timestep took approximately a week to simulate one hour of reactor time.

The results of the simulations with a 10 by 10 cell density and a times step of 0.1 milliseconds are presented. The results are presented for different cases such as: a case where a proton accelerator was coupled to a reactor; a case where the multiplicity of the reactors were subcritical (close to critical, k_{eff}= 0.974), subcritical (k_{eff}= 0.638), supercritical (k_{eff}= 1.368); a case where an electron accelerator was coupled to a reactor; and a case where the accelerator was shutdown after half an hour of runtime and the concentration of the poisons were monitored.

7.4.1 Proton Accelerator Coupled to a Reactor

Results presented are from reactors coupled to a 1000 MeV proton accelerator. Spatial distributions and time dependent behavior are presented separately in the following sections.

7.4.1.1 Spatial Distribution of Neutron Flux and Neutron Poisons in the Reactor

The results are presented from a nuclear reactor having k_{eff}=0.974 coupled to a 1000 MeV proton accelerator. Spatial distribution of neutron flux for five times steps and neutron poisons' spatial distribution are presented. Source neutron flux generated by the proton beam when the accelerator was "on" is shown in figure 7.18. The source neutron flux was maximum in the cells located near the target where the particle beam was impinged and it reduced gradually towards the edges. As expected, the neutron flux was maximum at the center of the reactor and gradually reduced towards the edges. Neutron

poisons concentration was maximum at the center of the reactor and reduced gradually towards the edges. As the poison was produced from fission reactions and fission reactions were proportional to neutron flux, it could be expected that the neutron poisons' distribution would be similar to the neutron flux distribution. As seen in figure 7.24 and 7.25, the neutron poison distribution was maximum at the center of the reactor and decreased gradually towards the edges, as expected.

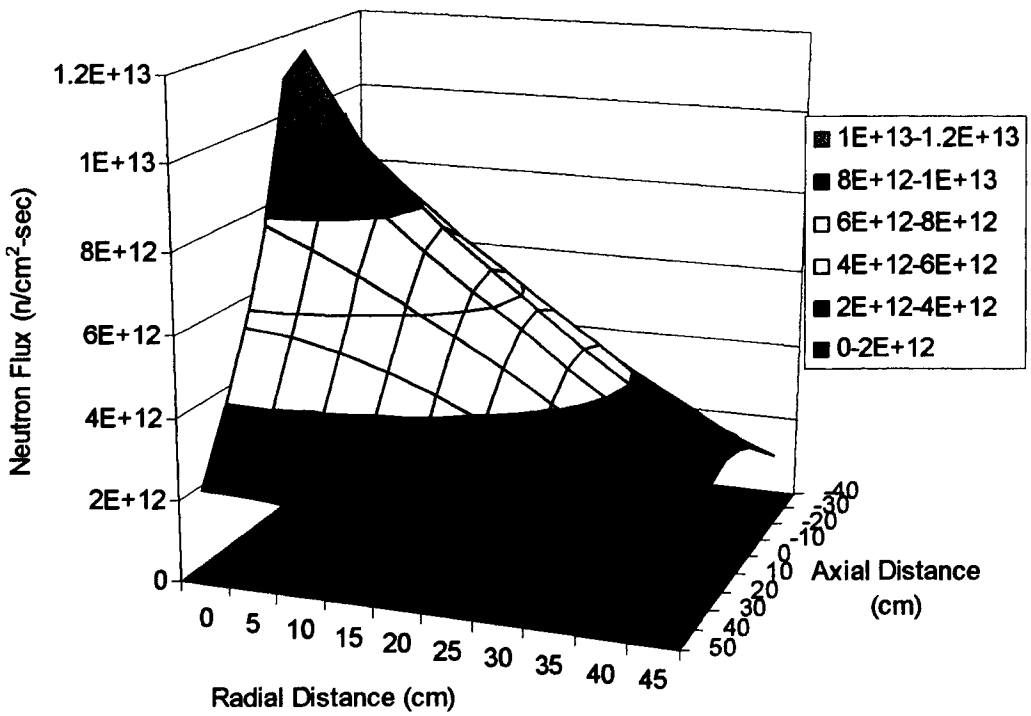


Figure 7.18 Source Neutron Flux Generated from a 1000 MeV Proton Beam

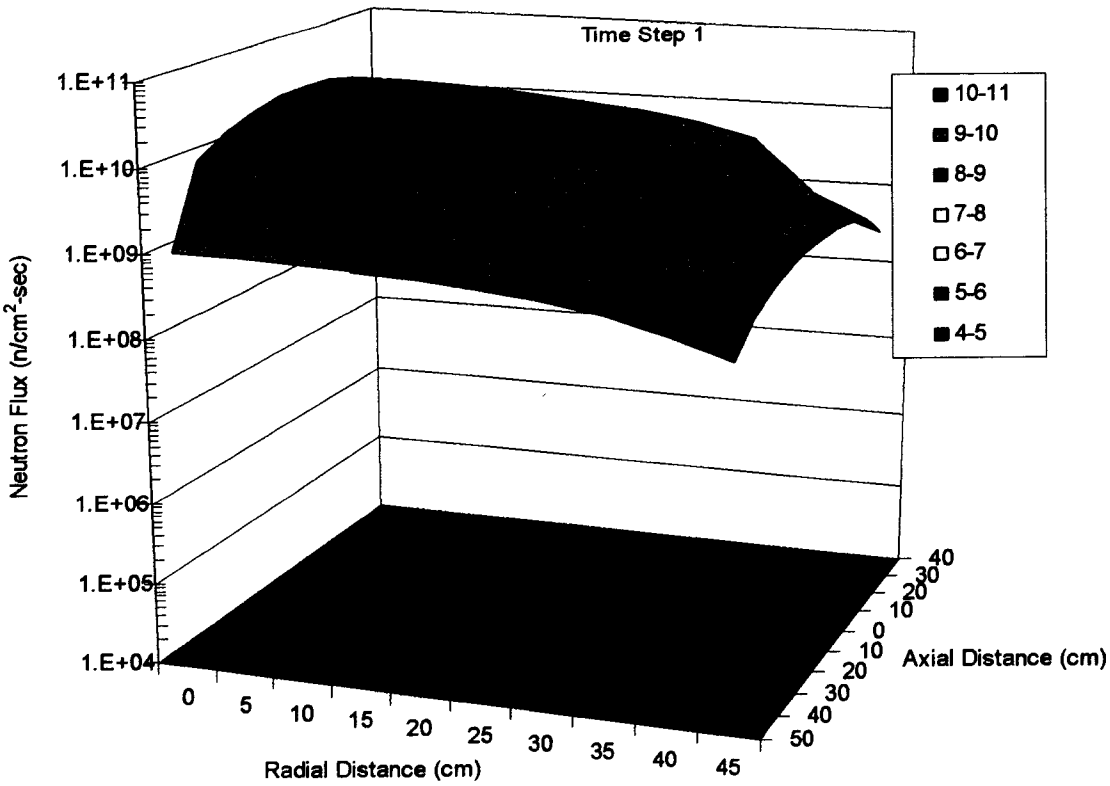


Figure 7.19 Neutron Flux Distribution in the Reactor at First Timestep

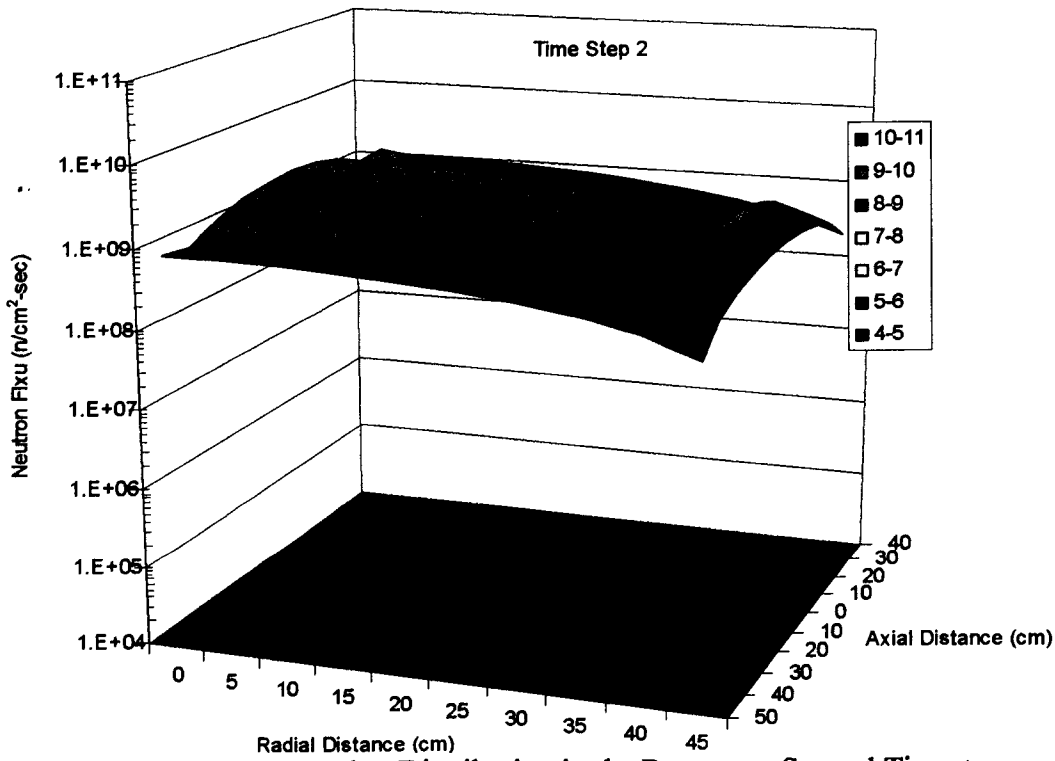


Figure 7.20 Neutron Flux Distribution in the Reactor at Second Timestep

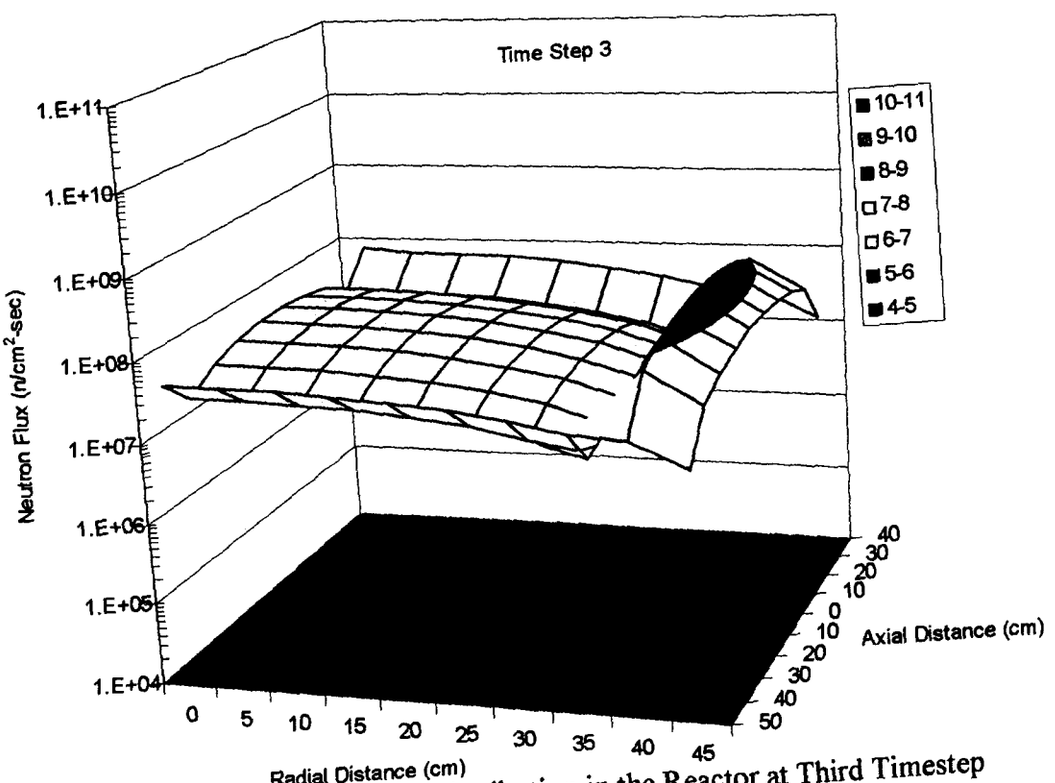


Figure 7.21 Neutron Flux Distribution in the Reactor at Third Timestep

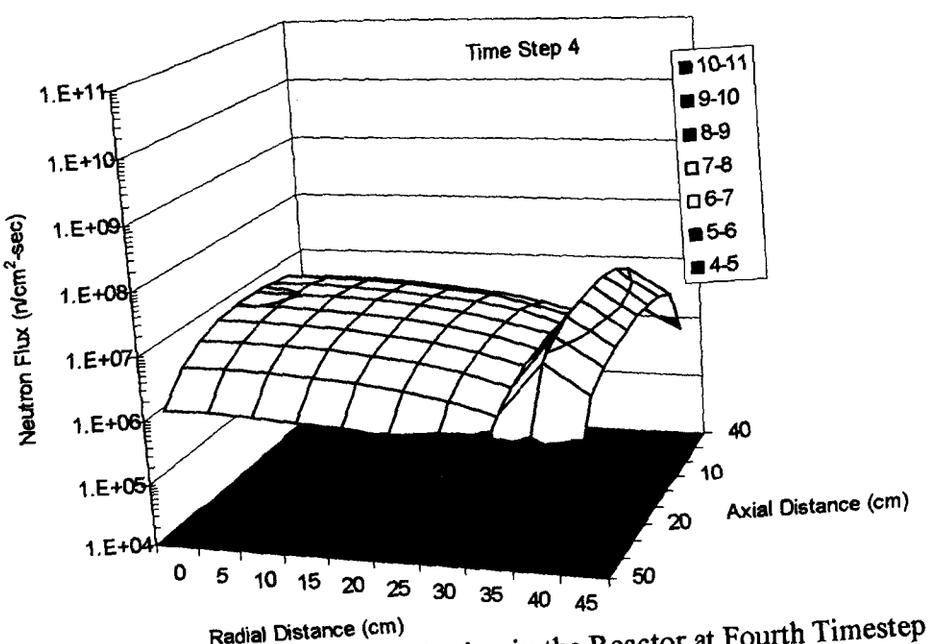


Figure 7.22 Neutron Flux Distribution in the Reactor at Fourth Timestep

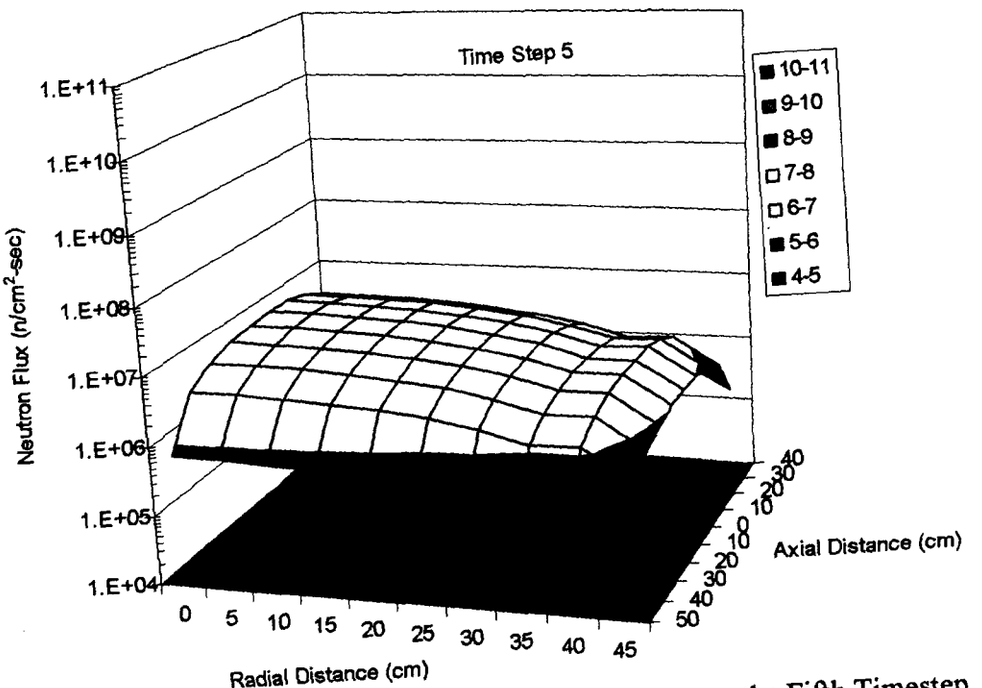


Figure 7.23 Neutron Flux Distribution in the Reactor at the Fifth Timestep

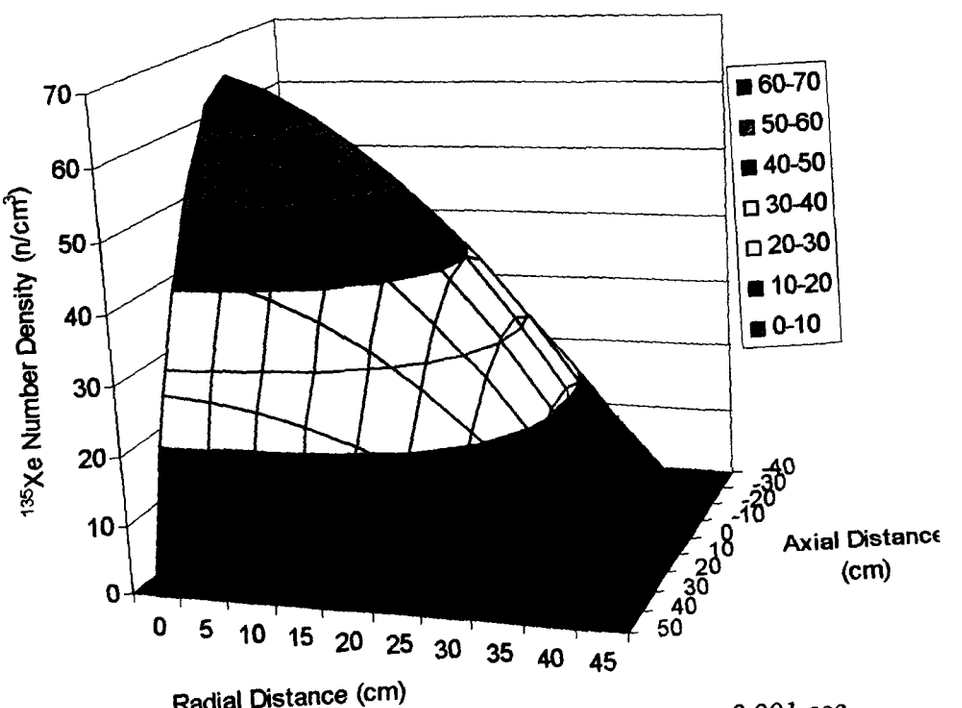


Figure 7.24 ^{135}Xe Distribution in the Reactor at 0.001 sec

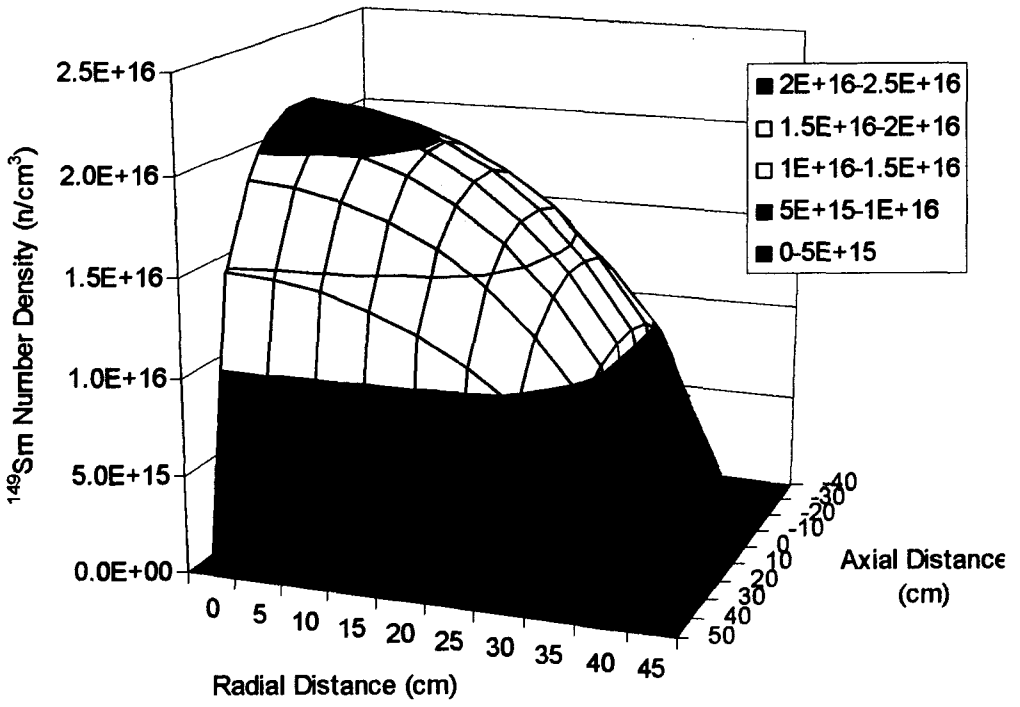


Figure 7.25 ^{149}Sm Distribution in the Reactor at 1400 sec

7.4.1.2 Time Dependent Results

Neutron flux variation at a particular location in the reactor over time, variation of concentration of neutron poison over time, and variation of plutonium number densities over time are presented for the case where a proton accelerator with specifications given above was coupled to a reactor having different multiplication factors (k_{eff}= 0.974, k_{eff}= 0.686, k_{eff}= 1.368). In all reactor cases, a 10% blanket and 90% moderating water, by volume, were considered. The percent of different minor actinides and fuel in the blanket are given in the table 7.8 for all cases. Important neutron poisons, such as samarium and xenon, vary over time, and the behavior of plutonium concentration over time is also presented for a reactor with k_{eff}= 0.974.

Table 7.8 Weight Percent of Actinides in the Blanket for Different Reactor Cases

k_{eff}	UO_2	PuO_2	NpO_2	^{241}Am	^{243}Cu
Subcritical (0.974)	73	2.5	2	20	2.5
Subcritical (0.686)	77	0.5	2	20	0.5
Supercritical (1.368)	88	4	2	2	4

Since the accelerators are pulsed, they fire the particles at regular intervals with a specific period. The general trend of time-dependent behavior of the neutron flux in the reactor in the short term was as expected for both subcritical cases. The neutron flux was maximum when the accelerator was fired and reduced gradually when the accelerator was off as shown in figures 7.27 and 7.30. In the supercritical case, the neutron flux also followed the similar trend as subcritical cases as shown in figure 7.33. This was due to the variations in the cross-sections from MCNPX to ADSTRANS, lack of delayed neutrons in the case of MCNPX and limited number of energy groups in the ADSTRANS. The time-dependent behavior of neutron flux in the reactors in long term (0.5 sec) was as expected for all the three cases. The increase of neutron flux in the supercritical case was faster than the subcritical cases as shown in figures 7.26, 7.29 and 7.32. The neutron flux variation over long period of time (1 hour) was also as expected for all the three cases. The neutron flux continued to build for some time in the beginning when there was no poison and reduces gradually as the neutron poison in the reactor buildup and reaches a steady state as shown in figures 7.28, 7.31 and 7.34. In contrast, the neutron poisons continued to build for some time and reached a steady state where the production from fission and loss due to neutron absorption were balanced as shown in figures 7.35 and 7.36.

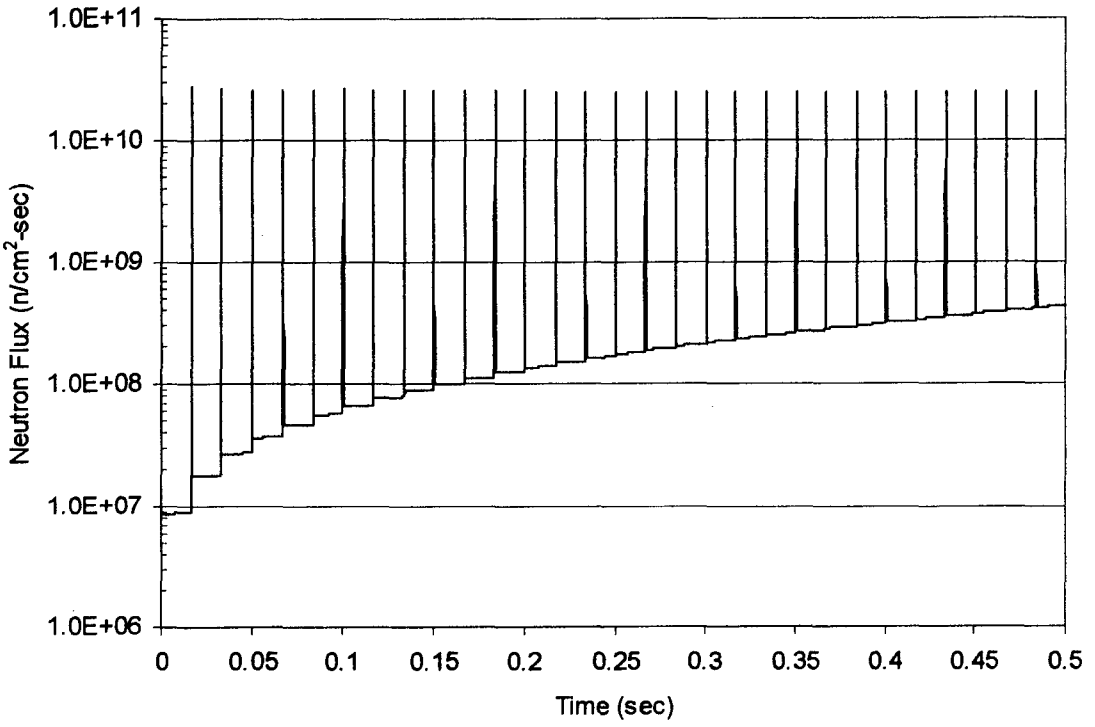


Figure 7.26 Neutron Flux Variation in the Reactor over Time for k_{eff}= 0.974

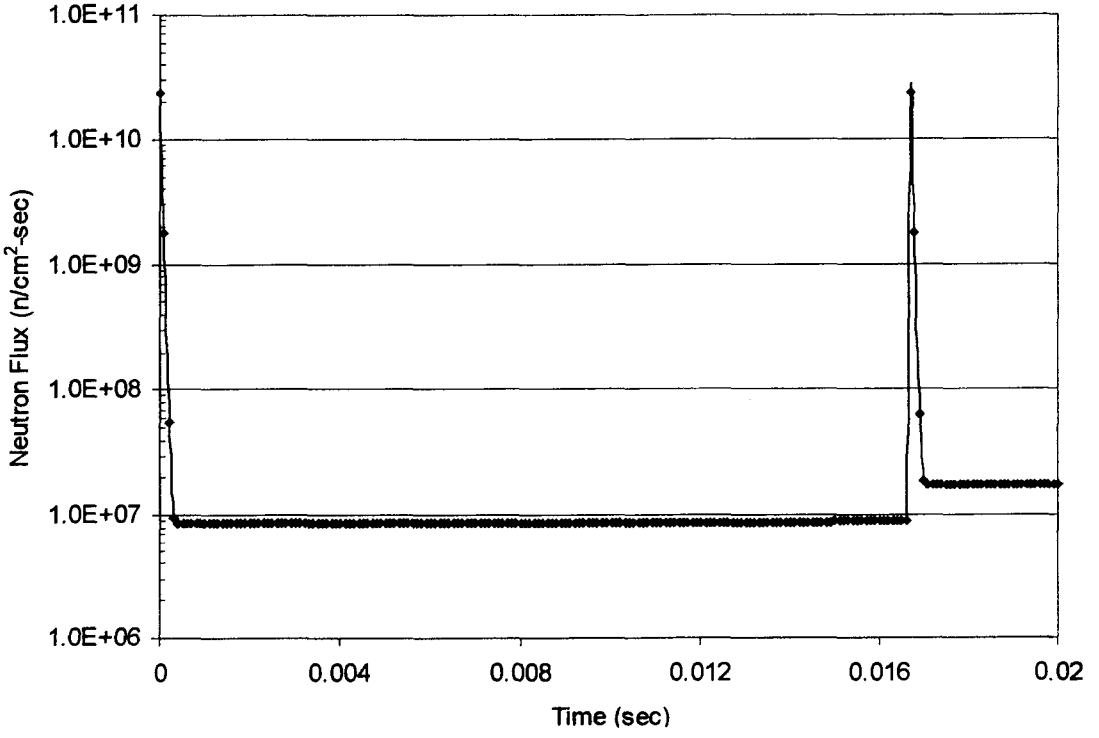


Figure 7.27 Neutron Flux Variation in the Reactor between Accelerator Firings

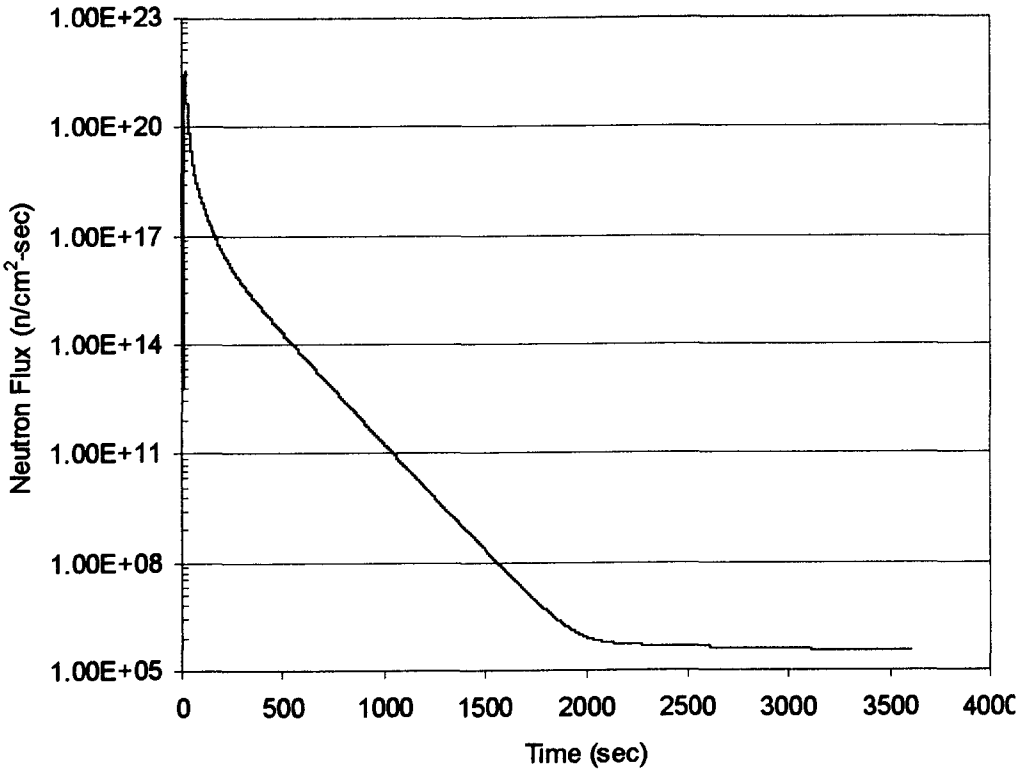


Fig. 7.28 Long Term Neutron Flux Variation in the Reactor for k_{eff}= 0.974

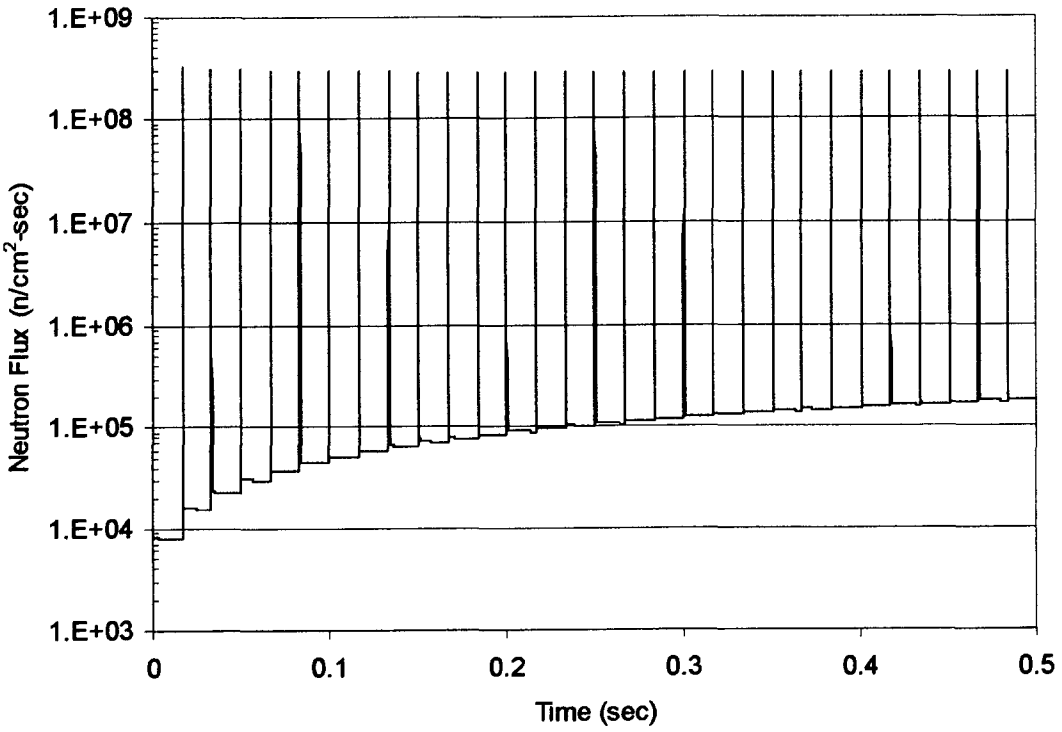


Figure 7.29 Short Term Neutron Flux Variation in the Reactor for k_{eff}= 0.686

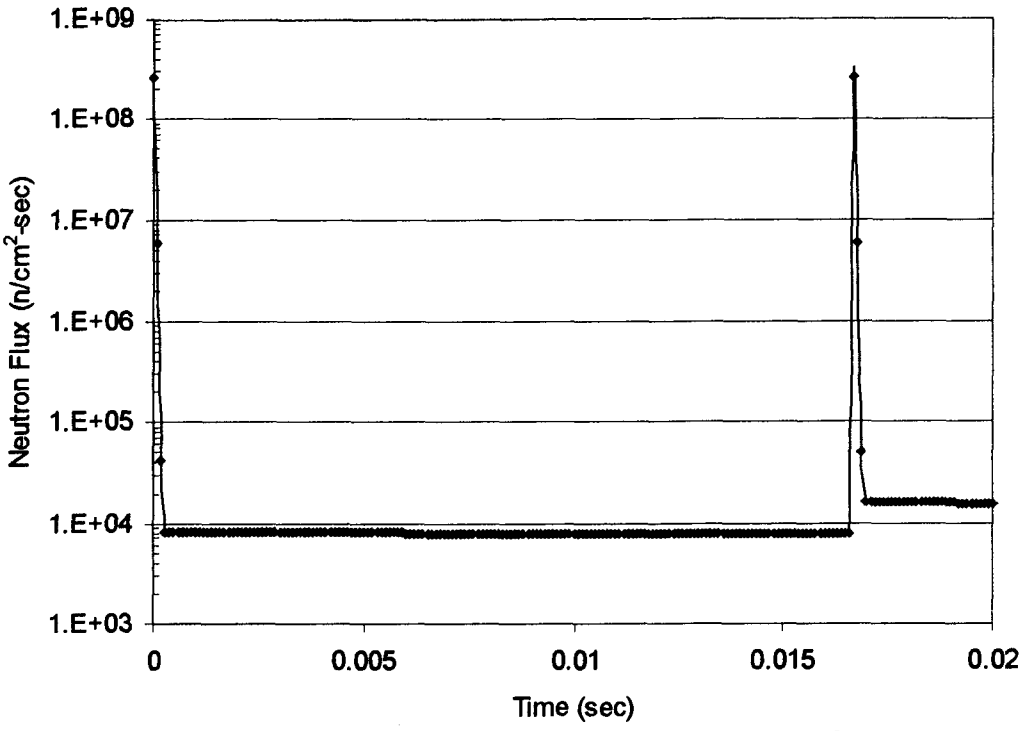


Figure 7.30 Short Term Neutron Flux Variation in the Reactor for k_{eff}= 0.686

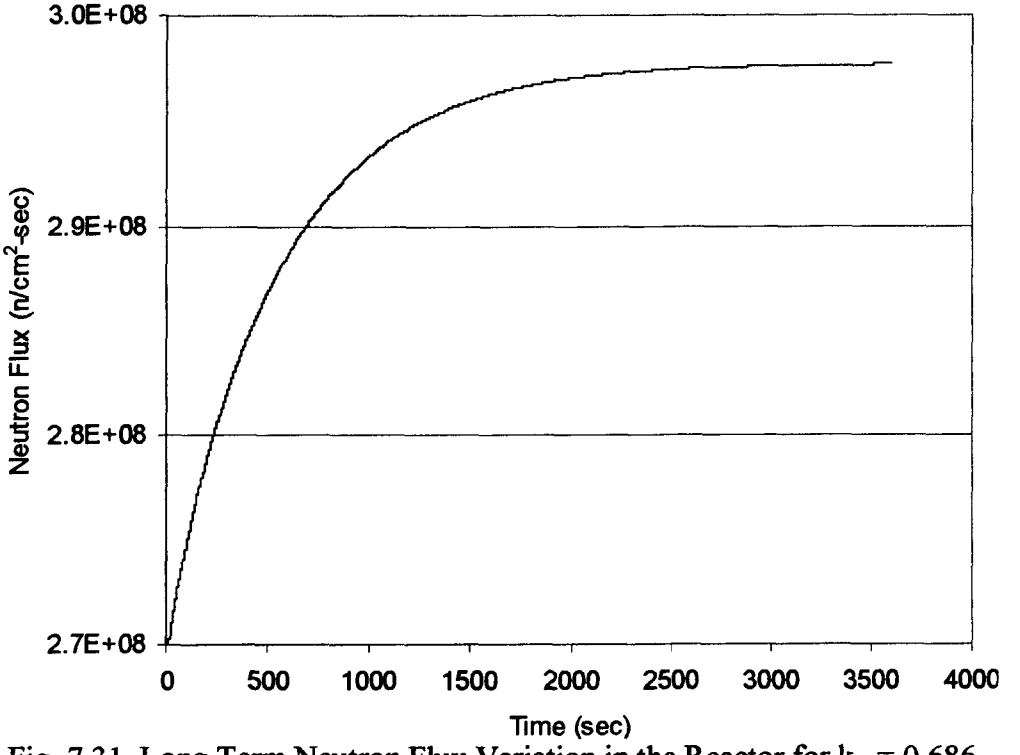


Fig. 7.31 Long Term Neutron Flux Variation in the Reactor for k_{eff}= 0.686

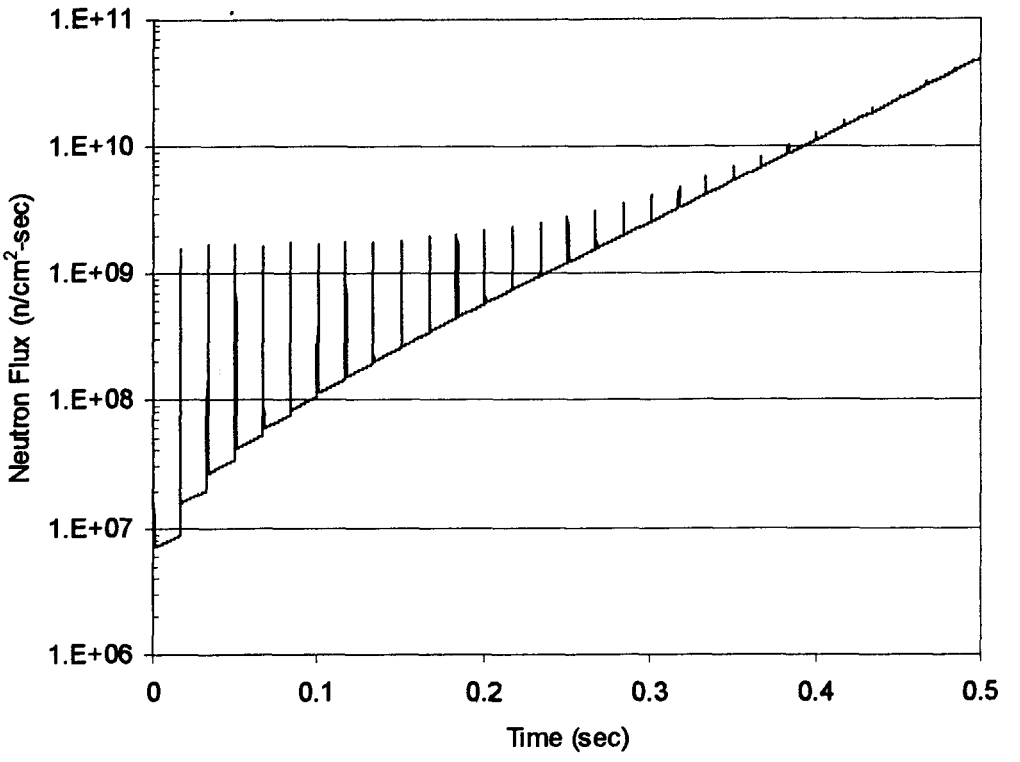


Figure 7.32 Neutron Flux Variation in the Reactor for k_{eff}=1.368

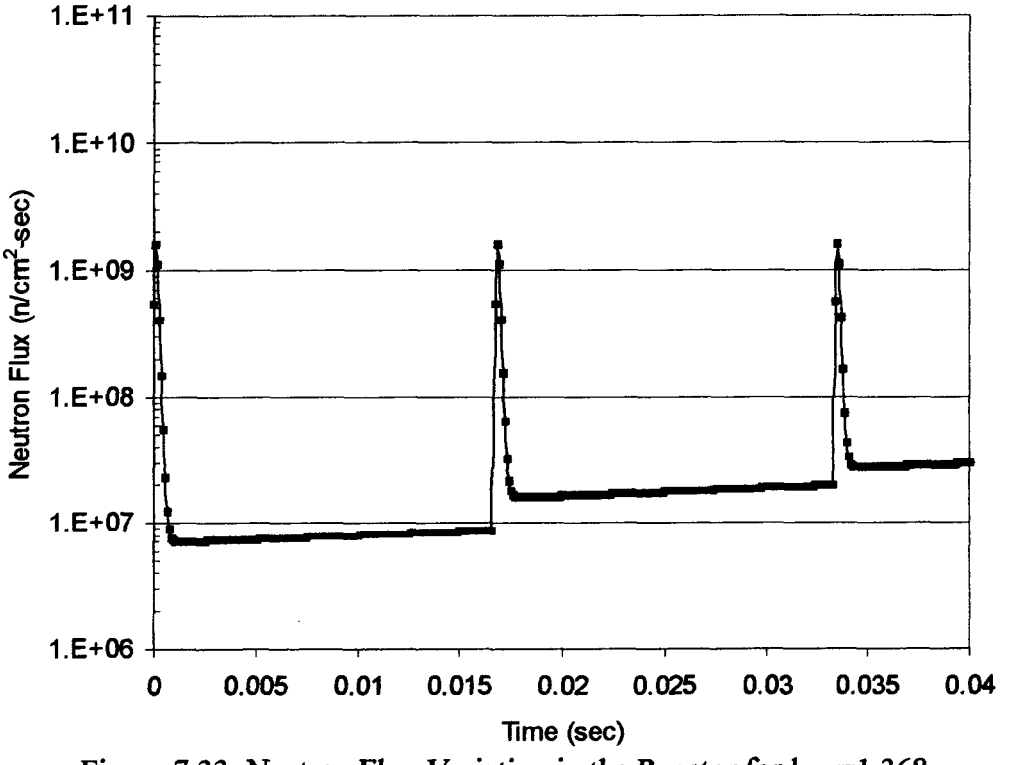


Figure 7.33 Neutron Flux Variation in the Reactor for k_{eff}=1.368

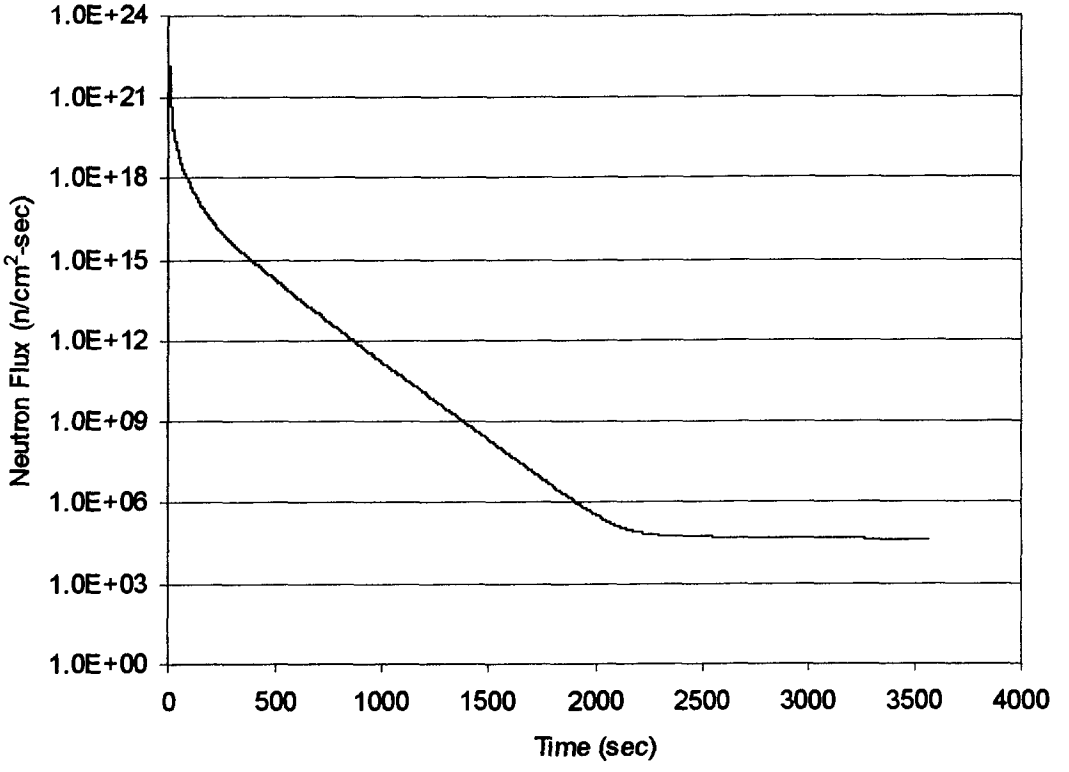


Figure 7.34 Long Term Neutron Flux Variation, k_{eff} =1.368

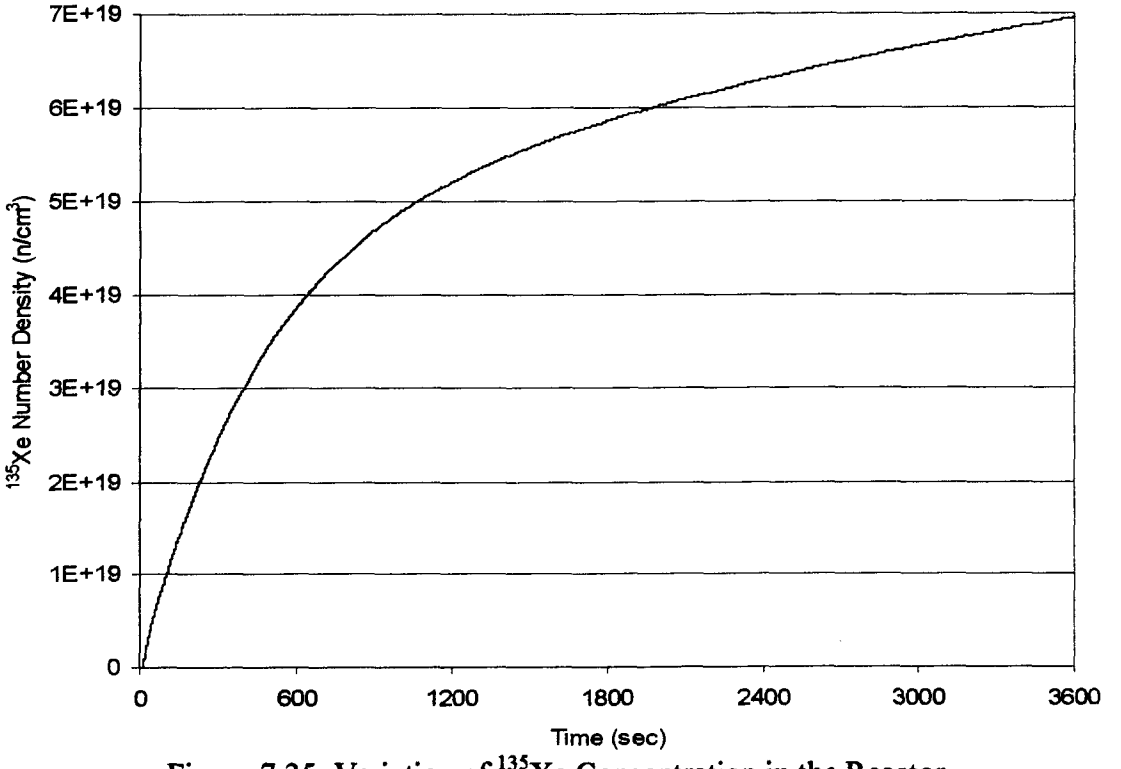


Figure 7.35 Variation of ^{135}Xe Concentration in the Reactor

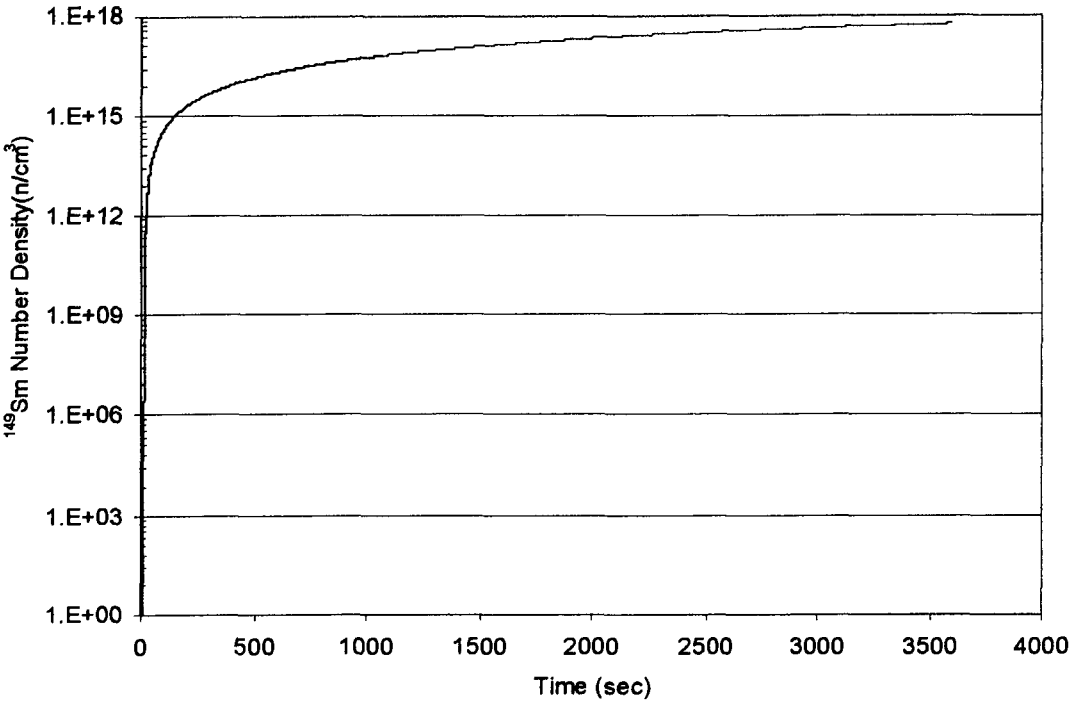


Figure 7.36 Variation of ^{149}Sm Concentration in the Reactor

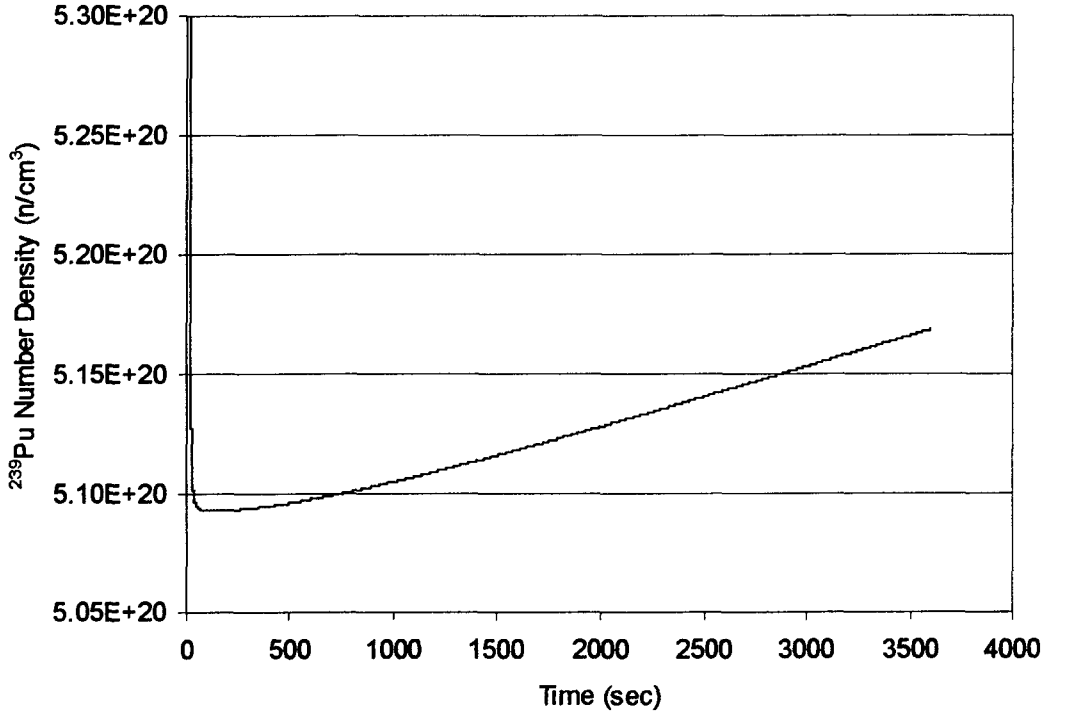


Figure 7.37 Variation of ^{239}Pu Concentration in the Reactor

7.4.2 Electron Accelerator Coupled to a Reactor

Neutron production from electrons was less when compared to protons as explained in Chapter 3. In the MCNPX simulations of neutron production from an electron beam, it was assumed that the whole reactor was filled with target material instead of core material. The [γ, n] cross-sections for most of the core materials were not available. Source neutron flux generated in the reactor from a 1000 MeV electron beam, when the accelerator was "on" is shown in figure 7.38. The source neutron flux is maximum in the cells located near the target where the particle beam is impinging and it reduced gradually towards the edges. Hence, the magnitude of neutron flux in an ADS is less from an electron accelerator compared to proton accelerator. However, the basic behavior of an ADS should be the same for electron or proton accelerator. Time dependent variation of neutron flux in a reactor coupled to an electron accelerator is presented. As can be expected, the neutron flux variation over time is similar to the one predicted for a proton accelerator as shown in figures 7.39 and 7.41.

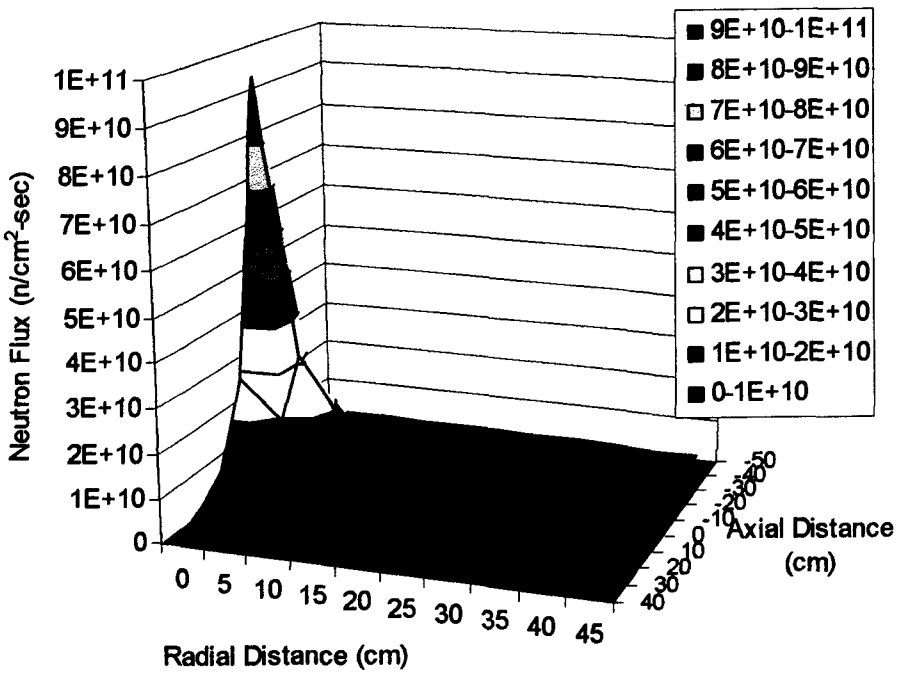


Figure 7.38 Neutron Flux Generated in the Reactor from a 1000 MeV Electron Beam

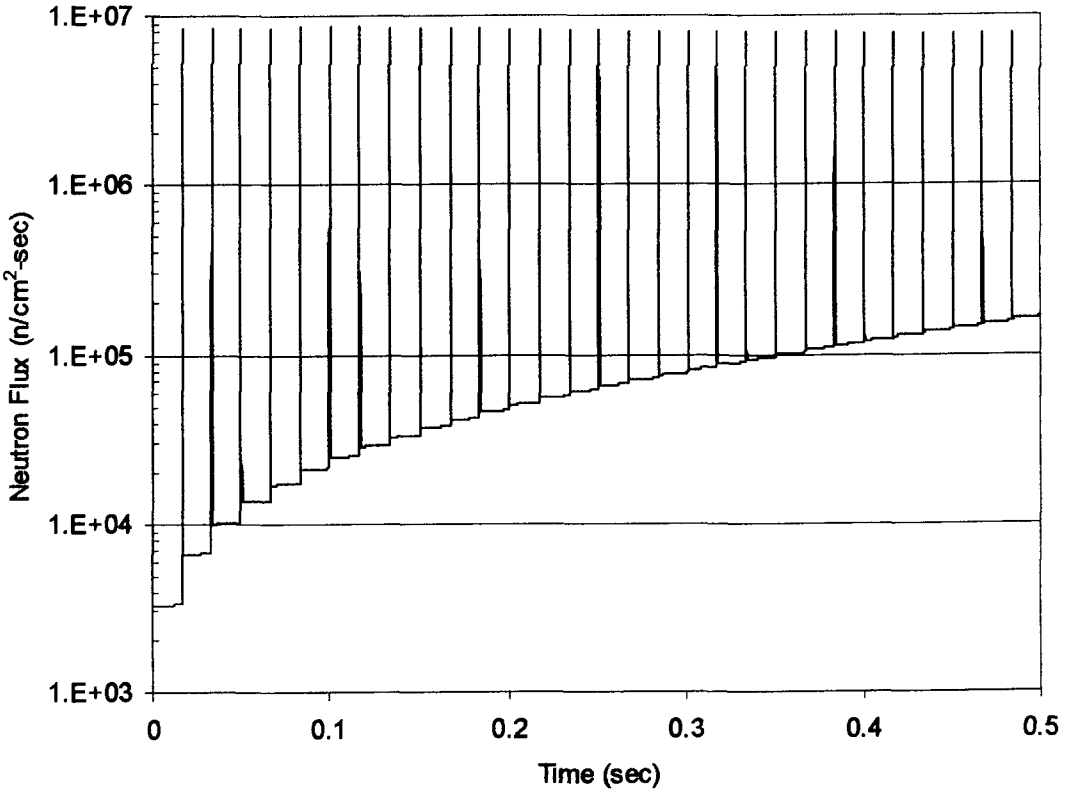


Figure 7.39 Short Term Neutron Flux Variation in the Reactor for k_{eff}= 0.974, Coupled to an Electron Accelerator

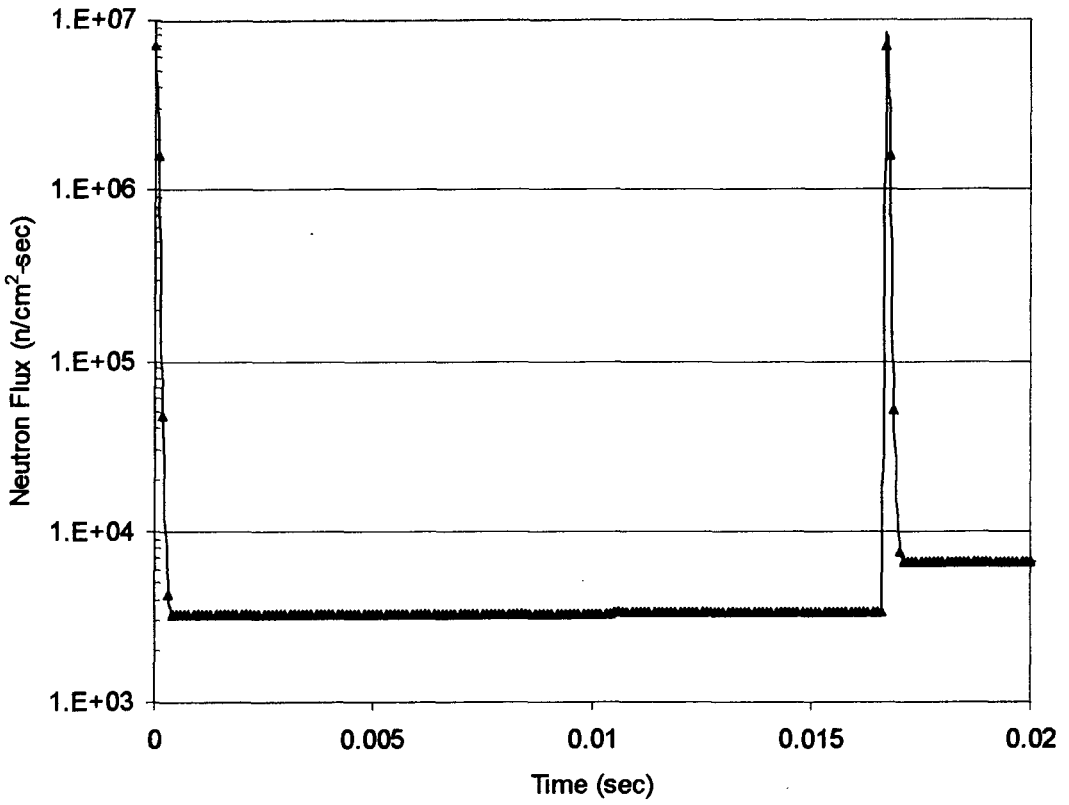


Figure 7.40 Short Term Neutron Flux Variation in the Reactor for k_{eff}= 0.974, Coupled to an Electron Accelerator

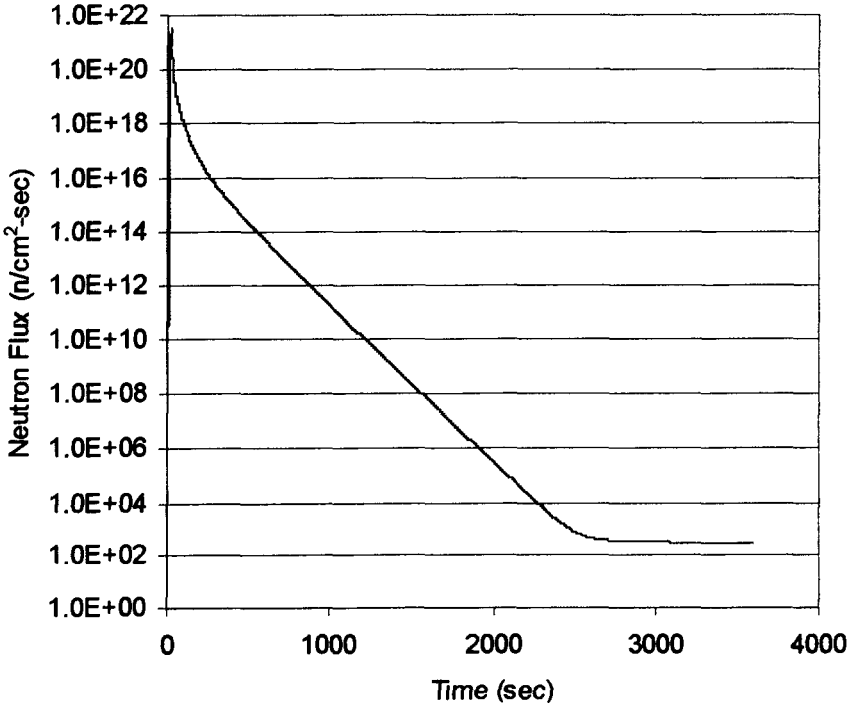


Figure 7.41 Long Term Neutron Flux Variation in the Reactor for k_{eff}=0.974, Coupled to an Electron Accelerator

7.4.3 Neutron Poisons Concentrations after the Reactor Shutdown

If the reactor is shut down after operating for a long period of time at a constant flux, the behavior of poison concentrations will be as explained in Sections 3.5.1 and 3.5.2 of Chapter 3. The behavior of ^{135}Xe and ^{149}Sm concentrations following reactor shutdown (after operating for 30 minutes) are presented in figures 7.42 and 7.43, respectively. The behavior of ^{135}Xe, ^{149}Sm concentrations are as expected as shown in the figures.

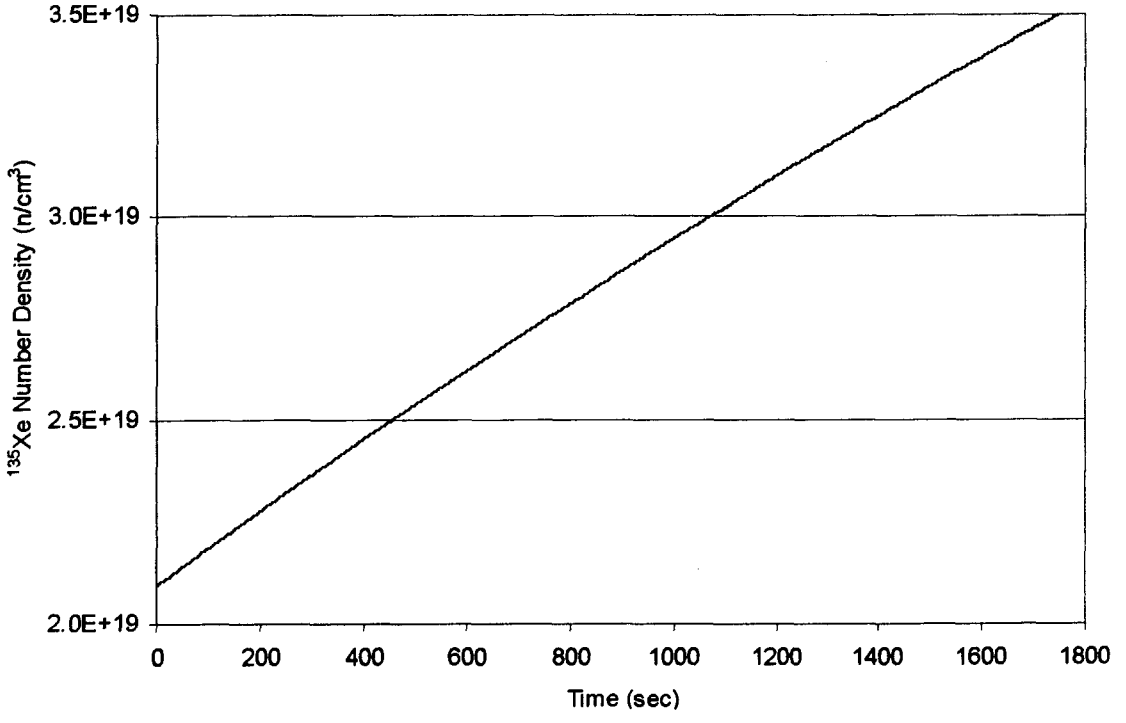


Figure 7.42 Behavior of ^{135}Xe Concentration Following the Reactor Shutdown, at t = 0

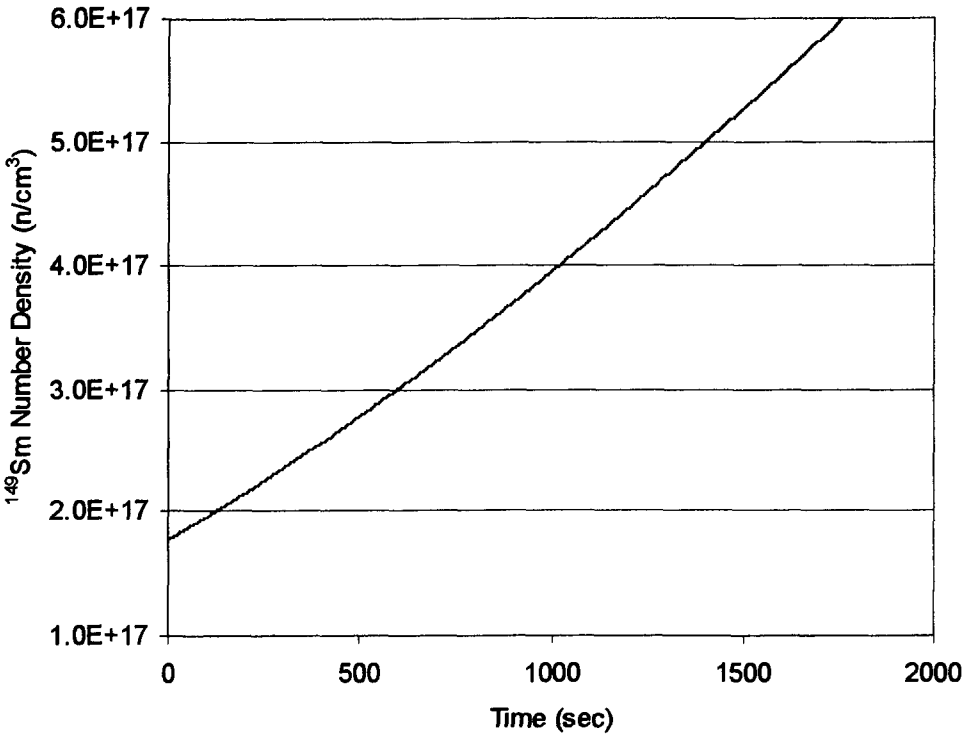


Figure 7.43 Behavior of ^{149}Sm Concentration Following Reactor Shutdown, at t = 0

7.4.4 Comparison of Neutron Flux in Reactors for k_{eff} = 0.686 and 0.974

Neutron flux in reactors with different neutron multiplication factors coupled to a proton accelerator predicted by ADSTRANS were compared. As it can be expected, neutron flux in a reactor with 0.974 neutron multiplication is higher than a reactor with 0.686 neutron multiplication as shown in the figure 7.44. Hence, the transmutation of actinides, which is proportional to neutron flux is more effective in a reactor with larger multiplication factor. However, due to the safety reasons the reactor in an ADS should be subcritical as suggested by many people.

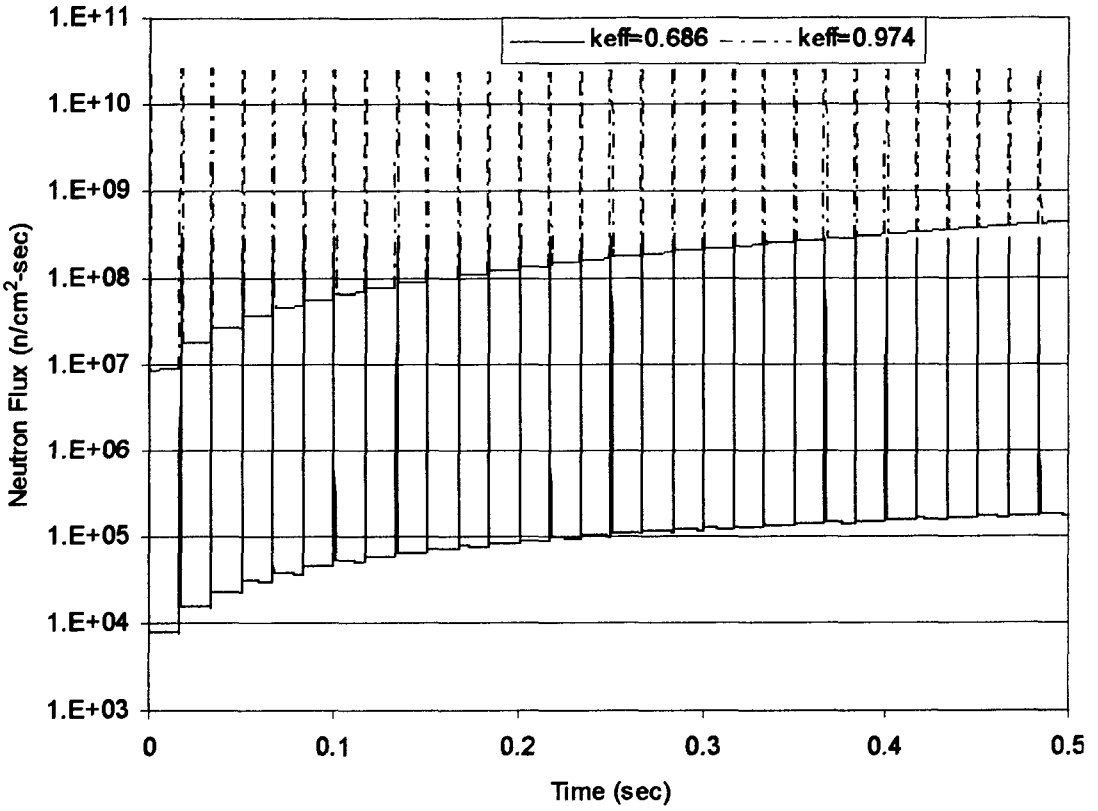


Figure 7.44 Comparison of Neutron Flux Variation in Reactors with k_{eff} = 0.686 and 0.974

7.4.5 Burnup of Minor Actinides (Pu, Am, and Cm) in the Transmuter

Results are presented for a case where the reactor is filled with 90% moderator (carbon) and 10% minor actinides. Minor actinides such as ^{239}Pu, ^{239}Np, ^{241}Am and ^{243}Cm were transmuted in the reactor by the neutrons produced from the accelerator and that were generated from the fission of ^{239}Pu. Since transmutation was proportional to the neutron flux in the reactor and the neutron flux was building up initially, the quantity of minor actinides transmuted was decreased. As seen from the figures 7.45, 7.46 and 7.47, the number densities of the minor actinides decreased over the time due to the transmutation process.

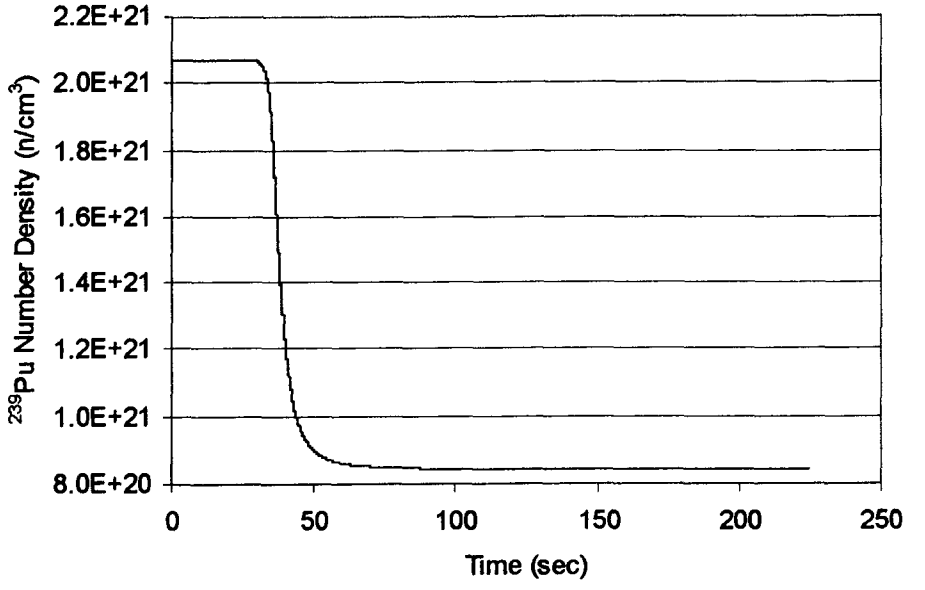


Figure 7.45 Plutonium Burnup in the Reactor Coupled to an Accelerator

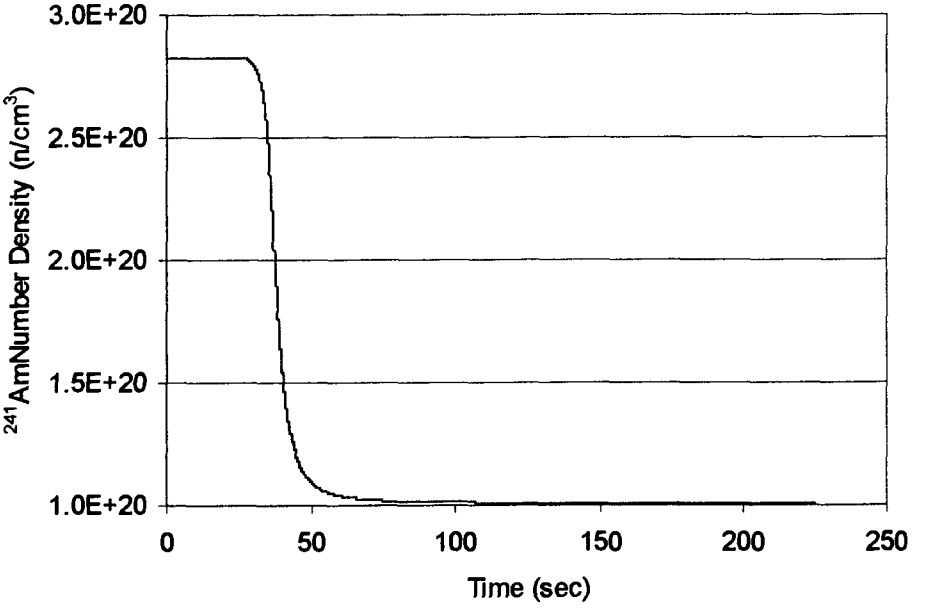


Figure 7.46 Americium Burnup in the Reactor Coupled to an Accelerator

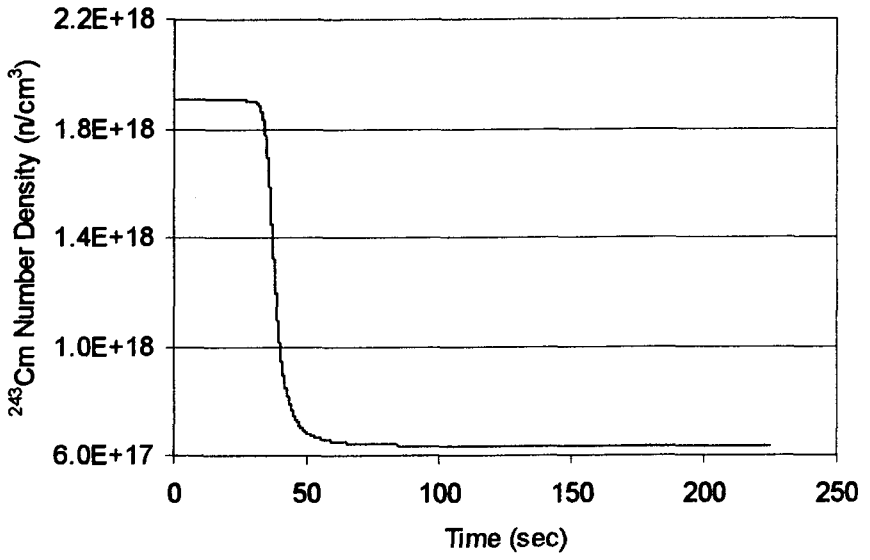


Figure 7.47 Curium Burnup in the Reactor Coupled to an Accelerator

CHAPTER 8

CONCLUSIONS AND COMMENTS

A two-dimensional, axisymmetric numerical code, "ADSTRANS," was developed to study neutron behavior, minor actinide burnup and neutron poison variation in a nuclear reactor coupled to an accelerator. The Monte-Carlo N-Particle transport code MCNPX, version 2.4.j, was used to predict the neutron production in a target from a particle beam produced by an accelerator.

8.1 Experimental Verification

MCNPX predictions were verified against experimental results. These benchmarking experiments were done at the Idaho Accelerator Center in Pocatello, Idaho, in 2002. Neutron production from a 20 MeV electron linear accelerator (LINAC) was compared with MCNPX predictions. The total neutron production predicted by MCNPX was within 10.9% of the experimental results providing good agreement. The IAC experiments verified that MCNPX and its cross-section database could be a powerful tool in the prediction of neutron production by photodisintegration. Colleagues within our research group were involved in similar experiments carried out on the LANSCE proton accelerator at the Los Alamos National Laboratory to validate the use of MCNPX on a

lead/bismuth spallation target. Once again, MCNPX results agreed well with the experimental data.

8.2 Verifications using Analytical Results

The "ADSTRANS" simulation results were also compared against analytical solutions. Three comparisons were conducted to test the ability of the ADSTRANS code to predict steady-state and transient behavior. These included:

1. Steady state, bare cylindrical core with axial symmetry.
2. Steady state, infinite cylindrical reactor with a radial reflector.
3. Transient, bare cylindrical reactor.

In all cases, the analytical solutions compared well with the numerical predictions. Predictions of neutron poison buildup caused by fission and radioactive decay were also compared with analytical solutions after reactor shutdown. These results were also very good. In summary:

- The spatial neutron flux distribution in a steady-state, finite, bare cylindrical reactor as predicted by the simulation was in good agreement with analytical results.
- The time-dependent neutron flux at a single node in a finite, bare cylindrical reactor predicted by the code agreed with the analytical calculations within 9.1%.
- The neutron flux predicted by the code at different radial distances in a steady state, infinite cylindrical reactor with a reflector were in good agreement with the analytical solutions, with in 4.2% deviation.

- The variation of xenon concentration at a node in the reactor over time, as predicted by the simulation, were 3.6% of the analytical results.

These comparisons verified the validity of ADSTRANS as a valid tool for the simulation of neutron behavior in nuclear reactors or transmuters.

8.3 Numerical Verifications

The ADSTRANS code was also verified for numerical stability. The time-dependent neutron flux in an accelerator-driven system (ADS) with different cell densities, including 10 x10 and 20 x 20, and with different timesteps of 0.1 milliseconds and 0.05 milliseconds, were compared.

- The spatial neutron flux distributions in a steady-state reactor as predicted by the code through the two different cell densities were in good agreement, within 12%.
- The time-dependent neutron fluxes in the reactor as predicted by the code with the two different cell densities were also in good agreement, within 10.5 %.
- The spatial neutron flux distributions in a steady-state reactor as predicted by the code with two different time steps were in good agreement.
- The time-dependent neutron fluxes in the reactor as predicted by the code through the two different time steps were also in good agreement.

These numerical comparisons verified the stability of the code at different grid spacing and iteration time interval.

Next, the results from a test case simulated by the code through 10 x 10 cell density and 0.1 milliseconds time step were analyzed. The spatial neutron flux distribution, time-dependent neutron flux variation at a node, minor actinide burnup and neutron poison

variation in the ADS were as expected. It can be concluded that ADSTRANS is a powerful tool in predicting the neutron behavior of a nuclear reactor coupled to an accelerator. ADSTRANS can be used for different particle types, accelerator specifications, target materials and reactor geometries and core materials. As can be seen from the long term variation of neutron flux, fission products should also be given serious attention in ADS usage.

8.4 Suggestions for Future Work

- The ADSTRANS simulation on a 10 x 10 mesh with a 0.1 milliseconds time step for one hour of reactor time took almost a week on a personal computer (specifications given in Chapter 7). The same simulation on a 20 x 20 mesh took almost a month to run just 500 seconds of reactor time. The code should be modified to allow parallel processing.
- ADSTRANS contains three energy groups and, hence, can be improved to accommodate more energy groups by making changes in the code and refining the cross-section database.
- To analyze an ADS with thermal transients, the code should be coupled with a thermal hydraulic model.

APPENDIX I

COMPUTER CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include "MATRIX.H"

/*****************************************
 *
 *  I.  Define Global Variables.
 *
*****************************************/

# define ISOTOPES 18   /*  Number of Isotopes in the Database. */
# define NUMBER_OF_ENERGY_GROUPS  3
              /*  Number of Neutron Cross Section  Energy Groups. */


#define reactor_dimension_R 50
#define reactor_dimension_Z 100

#define reactor_R_cells 10
#define reactor_Z_cells 10

#define N 100

 double pi = 3.14159;
 double Avagadros_number = 6.023e23;
 double mass_neutron = 1.6749544e-27;
 double percent_enrichment;

 double reactor_temperature,particle_energy,accelerator_current;
 double accelerator_frequency,accelerator_period;
 double accelerator_pulse_charge,accelerator_duty_cycle;
 int    accelerator_particle,target_material;
 int    Simulation_particles =10000;

 double particle_charge_electron=1.6e-19,particle_charge_proton= 1.6e-19;
 double particles_per_pulse;
 double Neutronfluence_perparticle[N][NUMBER_OF_ENERGY_GROUPS];
 double Source_Neutronfluence[N][NUMBER_OF_ENERGY_GROUPS];
 double speed[NUMBER_OF_ENERGY_GROUPS];
```

```
double Ave_energy_group[NUMBER_OF_ENERGY_GROUPS];
 double Source_flux[N][NUMBER_OF_ENERGY_GROUPS];

 int   number_x_cells,number_y_cells,number_z_cells;

 double keff, std_keff, Neutrons_produced;
 double tolerance = 0.1;
 double time_last_fired = 0.0,time_last_writen=0.0,time_last_writen1=0.0,
        write =0, write2 =0,time_last_writen2=0.0;
 double thickness_shield=10;

 char key;

 int junk;
 int bias = 0;
 int i,j,k,Row,Col,L,count,Circum_cell[50];
 int center_cell_z;

 double dr,dz,r[N];
 double A[N][N], phi[N],flux[N][NUMBER_OF_ENERGY_GROUPS],
        Total_flux[N], Total_S[N],temp, fission_source[N],
        delayed_source[N][6][NUMBER_OF_ENERGY_GROUPS], total_dealyed_sorce[N][6],
        [N][NUMBER_OF_ENERGY_GROUPS], B[N],
        D[N][NUMBER_OF_ENERGY_GROUPS],
        diffusion_length[N][NUMBER_OF_ENERGY_GROUPS];
double  N_Sb[N], N_Te[N], N_I[N], N_Xem[N], N_Xe[N],
        N_Nd[N], N_Pm[N], N_Sm[N];

/* Percent Enrichment is the Enrichment in the Uranium, Percent_oxides is the Composition of Metal
   Oxides in Percent by Weight, Percent U,Pu,Np are the Urnium, Plutonium, Neptunium Percents in the
   Oxides, Percent H2O is the Water Percent in the  Core   (Core + H2O  = 1, UO2+ PuO2 + NpO2 +
   Oher MA = 1)  */

  double percent_enrichment = 0.007;

  /* Percent by Volume the Core Material and Water */

  double percent_core = 0.90, percent_H2O = 0.1;

  /* Percent of Each Isotope in the Core */

  double percent_Cm_core =0.025, percent_Am_core=0.2,percent_UO2_core=0.73,
         percent_PuO2_core=0.025, percent_NpO2_core=0.02;

  double atomic_weight_U, molecular_weight_UO2, molecular_weight_PuO2,
         molecular_weight_NpO2,  molecular_weight_H2O, density_H2O;

  double percent_U_UO2, percent_Pu_PuO2,percent_Np_NpO2, percent_H_H2O;

  double density_core = 10.97,density_H2O = 1,density_reactor;

  double density_core_reactor,  density_H2O_reactor, density_Cm_reactor, density_Am_reactor,
         density_UO2_reactor, density_PuO2_reactor, density_NpO2_reactor;

  double density_U_reactor, density_Pu_reactor,density_Np_reactor,density_O2_reactor;
```

```
double density_U235_reactor,density_U238_reactor, density_H_reactor;

double percent_U235_reactor,percent_U238_reactor, percent_Pu_reactor, percent_Np_reactor,
        percent_Am_reactor,  percent_Cm_reactor,
        percent_O2_reactor, percent_H_reactor;

double N_UO2[N], N_U235[N], N_U238[N],N_PuO2[N],N_NpO2[N], N_H2O[N],
        N_O[N], N_Pu239[N],N_Np237[N], N_H[N],N_Am241[N],N_Cm243[N];

double Sigma_s[N][NUMBER_OF_ENERGY_GROUPS],
       Sigma_a[N][NUMBER_OF_ENERGY_GROUPS],
       Sigma_f[N][NUMBER_OF_ENERGY_GROUPS],
       Sigma_t[N][NUMBER_OF_ENERGY_GROUPS],
        Sigma_tr[N][NUMBER_OF_ENERGY_GROUPS],
        Sigma_r[N][NUMBER_OF_ENERGY_GROUPS],
        Sigma_s1to0[N],Sigma_s2to1[N];

double atomic_weight_Fe,density_Fe,N_Fe,A_Fe=56,percent_weight_Fe,
        atomic_weight_Cr,density_Cr,N_Cr,A_Cr=52,percent_weight_Cr;

double N_U239[N],N_Np239[N];

char input_Neutron[132];

time_t start,end;
double elapsed,check;


/*****************************************************
 *
 *  I.1  Define Global Variables Related to Nuclear
 *       Properties.
 *
*****************************************************/

struct DATA {
  int Z;
  int A;
  int ID;
  char name[20];
  char isotope[6];
  double Aw;
  double density;
  double sigma_c[NUMBER_OF_ENERGY_GROUPS];
  double sigma_s[NUMBER_OF_ENERGY_GROUPS];
  double sigma_f[NUMBER_OF_ENERGY_GROUPS];
  double sigma_a[NUMBER_OF_ENERGY_GROUPS];
  double sigma_s1to0;
  double sigma_s2to1;
  double nu[NUMBER_OF_ENERGY_GROUPS];
  double sigma_t[NUMBER_OF_ENERGY_GROUPS];
} nuclear[ISOTOPES];

struct {
  double r_min;
  double r_max;
```

```
   double z_min;
   double z_max;
 } cell[reactor_R_cells][reactor_Z_cells];


 /*********************************************
  *
  * I.2 Define Functions.
  *
 *********************************************/


     int get_setupdata();
     int get_properties();
     double cell_dimensions();
     int make_mcnpx_keff1();
     int make_mcnpx_input1();
     int make_mcnpx_input2();
     void source();
     double Read_Neutron1();
     double Read_Neutron2();
     double Read_keff();

main () {

double t;
double time_step = 0.0001, maximum_time = 120;

FILE *fp2, *fp3, *fp_mcnpx, *fp4, *fp5, *fp6, *fp7,
      *fp8, *fp9,*fp10,*fp11,*fp12, *fp13, *fp_results;

int Shield_Row,Shield_Col;

double  f[NUMBER_OF_ENERGY_GROUPS],  X[NUMBER_OF_ENERGY_GROUPS];

double  fission_yield_Sb=0.00638,  fission_yield_Te=0.012 ,  fission_yield_I=0.036,
        fission_yield_Xem=0.00638,fission_yield_Xe=0.00228,fission_yield_Nd=0.0113;

double decay_const_Sb, decay_const_Te, decay_const_I, decay_const_Xem,
        decay_const_Xe, decay_const_Nd, decay_const_Pm;

double  decay_const_U239, decay_const_Np239;

double C[N][6],delayed_neutron[N],
        In_Scatter_neutrons[N][NUMBER_OF_ENERGY_GROUPS],
        Sum_Beta[N][NUMBER_OF_ENERGY_GROUPS][ISOTOPES],
        decay_const_C[6], decay_halflife_C[6];


/* Neutron Yield From the Precursor Groups (Pu-239) */

double Beta[6],beta[N][6][NUMBER_OF_ENERGY_GROUPS];
double total_delayed_yield[ISOTOPES];

/* Assigning the Total Dealyed Neutron Yield for Different Isotopes */
```

```
  total_delayed_yield[2] = 0.01668; total_delayed_yield[0] = 0.0;      total_delayed_yield[9] = 0.0;
  total_delayed_yield[3] = 0.04600; total_delayed_yield[1] = 0.0; total_delayed_yield[10] = 0.0;
  total_delayed_yield[4] = 0.00645; total_delayed_yield[5] = 0.0; total_delayed_yield[11] = 0.0;
  total_delayed_yield[14] = 0.001668; total_delayed_yield[6] = 0.0; total_delayed_yield[12] = 0.0;
  total_delayed_yield[17] = 0.001668; total_delayed_yield[7] = 0.0; total_delayed_yield[13] = 0.0;
  total_delayed_yield[15] = 0.00645; total_delayed_yield[8] = 0.0; total_delayed_yield[16] = 0.0;

  /* Calculating the Center Cell of the Reactor in the Axial Direction
      Where the Target is Placed */

  center_cell_z = (reactor_Z_cells/2) - 1;


  Beta[0] = 0.038, Beta[1] = 0.280, Beta[2] = 0.216, Beta[3] =0.328,
  Beta[4] = 0.103, Beta[5] = 0.035;

/* Precursor Group Decay constants of U235 */
  decay_halflife_C[0]=54.51, decay_halflife_C[1]=21.84, decay_halflife_C[2]=6.00,
  decay_halflife_C[3]=2.23, decay_halflife_C[4]=0.496, decay_halflife_C[5]=0.179;

/* Fraction of the Delayed Neutrons Appearing in Each Group */
  f[0] = 0, f[1] = 0, f[2] = 1;

/* Fraction of the Delayed Neutrons Appearing in Each Group */
  X[0] = 0.65, X[1] = 0.3, X[2] = 0.05;


/* Initializing the Number Densities of Precursor Groups */

   for(L=0; L<N; L++) {
    for (i=0; i<6; i++) {
      C[L][i] =0;
    }
   }


/* Initializing the Number Densities for Plutonium and Neptuniun */
  for(L=0; L<N; ++L) {
    N_U239[L] =0, N_Np239[L] =0;
  }

  for(i=0;i<6;i++) {
   decay_const_C[i] = log(2.00)/decay_halflife_C[i];
  printf("decay_const_C[%d] = %lf\n", i, decay_const_C[i]);
  }


/*scanf("%s", key);*/

/* Initial Condition of the Poisions */

 for(L=0;L<N;L++){
     N_Sb[L]= N_Te[L]= N_I[L]= N_Xem[L]= N_Xe[L]=
     N_Nd[L]= N_Pm[L]= N_Sm[L]= 0;
  }
/* Initialising Fission Source */
```

```
for(L=0; L<N; L++)  {
   fission_source[L] = 0;
}



double yield_Sb=0.05,    yield_Te=0.05, yield_I=0.06386, yield_Xem=0.05,
       yield_Xe=0.00228, yield_Nd=0.05;

/* Decay Half Life of Poisons in Sec */
double decay_half_Sb=1.7, decay_half_Te=19.2, decay_half_I=23688,
        decay_half_Xem=918, decay_half_Xe=33012,
        decay_half_Nd=7200, decay_half_Pm=194400;

double decay_half_U239=1380, decay_half_Np239=207360;

/* Calculating the Decay Constants of Poisons */

 decay_const_Sb= log(2.0)/decay_half_Sb;
 decay_const_Te=log(2.0)/decay_half_Te;
 decay_const_I = log(2.0)/decay_half_I ;
 decay_const_Xem=log(2.0)/decay_half_Xem;
 decay_const_Xe= log(2.0)/decay_half_Xe;
 decay_const_Nd=log(2.0)/decay_half_Nd;
 decay_const_Pm= log(2.0)/decay_half_Pm;

 decay_const_U239 = log(2.0)/decay_half_U239;
 decay_const_Np239= log(2.0)/decay_half_Np239;


/************************************
*
*  A.  Input Data.
*
************************************/

  printf("************************************\n");
  printf("*  Program:  NUKETRANS       *\n");
  printf("************************************\n");
  printf("\n");
  printf("\n");



/************************************
    *
    *  A.1.  Obtain User Input from the
    *     File SETUP.DAT.
    *
 ************************************/

        get_setupdata();


 /************************************
  *
```

```
 *  A.2.  Input Nuclear Data from File
 *        FILE NUCLEAR.DAT.
 *
 ************************************/


        get_properties();


 /* Calculating Absorption and Total Cross Sections */

    for (i=0; i<ISOTOPES; i++) {
      for(k=0; k<NUMBER_OF_ENERGY_GROUPS; k++) {
       nuclear[i].sigma_a[k] = nuclear[i].sigma_c[k] + nuclear[i].sigma_f[k];
       nuclear[i].sigma_t[k] = nuclear[i].sigma_a[k] + nuclear[i].sigma_s[k] +      nuclear[i].sigma_s[k];

      }
    }


/************************************
*
*  B.  Calculate the Transient
*      Neutron Flux.
*
************************************/

   /*********************************
   *
   *  B.1 Calculating the Number
   *      Densities
   *
   *********************************/


  /* FeCr Reflector */
       atomic_weight_Fe = 55.85;
       density_Fe       = 7.86;
       percent_weight_Fe= 0.9;
       N_Fe = (percent_weight_Fe*density_Fe *   Avagadros_number)/atomic_weight_Fe;

       atomic_weight_Cr = 51.996;
       density_Cr       = 7.19;
       percent_weight_Cr= 0.1;
       N_Cr = (percent_weight_Cr*density_Cr * Avagadros_number)/atomic_weight_Cr;


  /* Uranium Core */
     atomic_weight_U = 1.00/((percent_enrichment/235.0439) +
                             ((1.00-percent_enrichment)/238.0508));

     molecular_weight_UO2 = atomic_weight_U + 2 * nuclear[1].Aw;
     molecular_weight_PuO2 = nuclear[4].Aw + 2 * nuclear[1].Aw;
     molecular_weight_NpO2 = nuclear[14].Aw + 2 * nuclear[1].Aw;
     molecular_weight_H2O = 2 * nuclear[0].Aw +  nuclear[1].Aw;
```

```
        percent_U_UO2  = atomic_weight_U /molecular_weight_UO2;
        percent_Pu_PuO2 = nuclear[4].Aw /molecular_weight_PuO2;
        percent_Np_NpO2 = nuclear[14].Aw /molecular_weight_NpO2;
        percent_H_H2O = (2 * nuclear[0].Aw) /molecular_weight_H2O;

        density_core_reactor = density_core * percent_core;
        density_H2O_reactor  = density_H2O * percent_H2O;
        density_reactor      = density_core_reactor + density_H2O_reactor;

        density_Cm_reactor    = density_core_reactor * percent_Cm_core;
        density_Am_reactor    = density_core_reactor * percent_Am_core;
        density_UO2_reactor   = density_core_reactor * percent_UO2_core;
        density_PuO2_reactor  = density_core_reactor * percent_PuO2_core;
        density_NpO2_reactor = density_core_reactor * percent_NpO2_core;

        density_U_reactor   = density_UO2_reactor * percent_U_UO2;
        density_Pu_reactor  = density_PuO2_reactor* percent_Pu_PuO2;
        density_Np_reactor  = density_NpO2_reactor* percent_Np_NpO2;

        density_O2_reactor  = (1-percent_U_UO2)*density_UO2_reactor
                            + (1-percent_Pu_PuO2)*density_PuO2_reactor
                            + (1-percent_Np_NpO2)*density_NpO2_reactor
                            + (1-percent_H_H2O)*density_H2O_reactor;

        density_U235_reactor = density_U_reactor * percent_enrichment;
        density_U238_reactor = density_U_reactor * (1.00-percent_enrichment);
        density_H_reactor    = density_H2O_reactor * percent_H_H2O;

    /* Percent Fractions of Each Isotope is Calculated, for Use in MCNPX files */
        percent_U235_reactor  = density_U235_reactor/density_reactor;
        percent_U238_reactor  = density_U238_reactor/density_reactor;
        percent_Pu_reactor    = density_Pu_reactor/density_reactor;
        percent_Np_reactor    = density_Np_reactor/density_reactor;
        percent_Am_reactor    = density_Am_reactor/density_reactor;
        percent_Cm_reactor    = density_Cm_reactor/density_reactor;
        percent_O2_reactor    = density_O2_reactor/density_reactor;
        percent_H_reactor     = density_H_reactor/density_reactor;

printf("atomic_weight_U = %lf\n", atomic_weight_U);
printf("atomic_weight_Pu = %lf\n", nuclear[4].Aw);
printf("molecular_weight_UO2 = %lf\n", molecular_weight_UO2);
printf("percent_U235_reactor  = %lf\n",percent_U235_reactor );
printf("percent_U238_reactor  = %lf\n",percent_U238_reactor );
printf("percent_Np_reactor = %lf\n", percent_Np_reactor);
printf("percent_Pu_reactor = %lf\n", percent_Pu_reactor);
printf("percent_Am_reactor = %lf\n", percent_Am_reactor);
printf("percent_Cm_reactor = %lf\n", percent_Cm_reactor);
printf("percent_O2_reactor =%lf\n", percent_O2_reactor);
printf("percent_H_reactor =%lf\n", percent_H_reactor);
printf("percent_core_reactor =%lf\n", percent_core);
printf("percent_H2O_reactor =%lf\n", percent_H2O);

printf("density_core_reactor = %lf\n", density_core_reactor);
printf("density_H2O_reactor = %lf\n", density_H2O_reactor);
```

```
/*************************************
*
* B.2 Dividing the Reactor into Finite
*     Cell and Storing their Dimensions
*
*************************************/

     cell_dimensions();

    for(L=0,i=0; i<reactor_R_cells; ++i)  {

       r[L] = (double)dr/2.00;

       for(j=0; j<reactor_Z_cells; ++j)  {
          r[L+1] = r[L]+ dr;
          L = L+1;

       }
    }


/*************************************
*
*  B.3  Run MCNPX to Obtain Neutron
*       Source Term.
*
*************************************/

  /*********************************
  *
  * B.3.A  Creating MCNPX Input File
  *
  *********************************/


    make_mcnpx_keff();
    make_mcnpx_Neutron1();


  /*********************************************
  *
  * B.3.B Running MCNPX Files, Reading the
  *       Required Information and Generating
  *       the Source Neutron Flux from Accelerator.
  *
  *********************************************/

     system("mcnpx.bat");


/* Reading the MCNPX Output File for Keff */

     Read_keff();

   printf("Keff = %lf  std_keff = %lf\n", keff, std_keff);
```

```
/* Reading the MCNPX Output File for Neutron Source Term from Accelerator */

    Read_Neutron2();


 /*********************************************
 *
 * B.3.C Calculating the Neutron Flux from
 *      Accelerator Specifications.
 *
 *********************************************/

    if(accelerator_particle==3||accelerator_particle==9 ) {
      particles_per_pulse = accelerator_pulse_charge/particle_charge_electron;
    } else {
        printf("The present phase of the code can only run electon or proton please   consider either of
                those and run again\n");
     }

     for(L=0; L<N; ++L) {
       for(k=0; k<NUMBER_OF_ENERGY_GROUPS; k++) {
         Source_Neutronfluence[L][k] = Neutronfluence_perparticle[L][k] * particles_per_pulse;
         Source_flux[L][k] = Source_Neutronfluence[L][k] * accelerator_frequency;
       }
     }


/*******************************************
*
* B.4 Assigning the Average Energies of Three
*    Energy Groups (Considered Thermal, Epithermal
*    Fast Groups Based on Plutonium Cross-Section Plots)
*
*******************************************/

  Ave_energy_group[0] = 0.569826;          /* Energy in ev */
  Ave_energy_group[1] = 3.1775e3;
  Ave_energy_group[2] = 20.3e6;

  /*********************************************************
  *
  * Calculating the Speed of Neutrons in Each Group (cm/s)
  *
  *********************************************************/

   for(k=0; k<NUMBER_OF_ENERGY_GROUPS; k++) {
    speed[k] = 1.383*pow(10,6)*sqrt(Ave_energy_group[k]);
   }


/*************************************************
*
* B.5 Calculating the Vaiables such as Number Densities
*       and Diffusion Coefficients in Each Cell
*
*************************************************/
```

```
/************************************************
*
 * B.5.A Assigning the Cell Indexes in Radial, Axial Directions
*          and Calculating the no. of Shield Rows, Columns
*
************************************************/

  Col = reactor_R_cells;
  Row = reactor_Z_cells;

/* Calculating the no. of Rows,Columns in the Shield Thickness*/

  Shield_Row = thickness_shield/dz;
  Shield_Col = thickness_shield/dr;


/*****************************************************
*
* B.5.B Calculating the Number Densities of Isotopes,
*         Cross-Sections and Diffustion Coefficeient etc.
*
*****************************************************/

 for(i=0,L=0; i<Row; ++i) {
   for(j=0; j<Col; ++j) {

    if((i<Shield_Row) || (i>=(Row-Shield_Row)) || (j>=(Col-Shield_Col)) ) {

     /* Steel Reflector Around the Reactor*/

      N_UO2[L]   = 0;
      N_U235[L]  = 0;
      N_U238[L]  = 0;
      N_PuO2[L]  = 0;
      N_Pu239[L] = 0;
      N_NpO2[L]  = 0;
      N_Np237[L] = 0;
      N_H2O[L]   = 0;
      N_O[L]     = 0;
      N_H[L]     = 0;
      N_Am241[L] = 0;
      N_Cm243[L] = 0;

for(k=0; k<NUMBER_OF_ENERGY_GROUPS; k++) {

 Sigma_a[L][k]  = N_Fe * nuclear[5].sigma_a[k] +
                  N_Cr *  nuclear[6].sigma_a[k];
 Sigma_f[L][k]  = N_Fe * nuclear[5].sigma_f[k] +
                  N_Cr * nuclear[6].sigma_f[k];
 Sigma_s[L][k]  = N_Fe * nuclear[5].sigma_s[k] +
                  N_Cr * nuclear[6].sigma_s[k];
 Sigma_t[L][k]  = (N_Fe * (nuclear[5].sigma_a[k] + nuclear[5].sigma_s[k])) +
                      (N_Cr * (nuclear[6].sigma_a[k] + nuclear[6].sigma_s[k]));
 Sigma_tr[L][k] = (N_Fe*nuclear[5].sigma_s[k]*(1-(2.00/(3.00*A_Fe))))+
                      (N_Cr*nuclear[6].sigma_s[k]*(1-(2.00/(3.00*A_Cr))));
```

```
    D[L][k] = 1.00/(3.00*Sigma_tr[L][k]);

    diffusion_length[L][k] = sqrt(D[L][k]/Sigma_a[L][k]);
    Sum_Beta[L][k][2]  = 0;
    Sum_Beta[L][k][3]  = 0;
    Sum_Beta[L][k][4]  = 0;
    Sum_Beta[L][k][14] = 0;
    Sum_Beta[L][k][17] = 0;
    Sum_Beta[L][k][15] = 0;

    fission_source[L]  = 0;
    }

/* Calculating the Group Transfer Cross Sections  in Shielding*/

    Sigma_s1to0[L]  = N_Fe * nuclear[5].sigma_s1to0 +
                      N_Cr * nuclear[6].sigma_s1to0;
    Sigma_s2to1[L]  = N_Fe * nuclear[5].sigma_s2to1 +
                      N_Cr * nuclear[6].sigma_s2to1;

 }
 else {

    N_UO2[L]    = (density_UO2_reactor * Avagadros_number)/
                  molecular_weight_UO2;
    N_U235[L]   = (density_U235_reactor * Avagadros_number)/
                  nuclear[2].Aw;
    N_U238[L]   = (density_U238_reactor * Avagadros_number)/
                   nuclear[3].Aw;
    N_PuO2[L]   = (density_PuO2_reactor * Avagadros_number)/
                  molecular_weight_PuO2;
    N_Pu239[L]  = N_PuO2[L];
    N_NpO2[L]   = (density_NpO2_reactor * Avagadros_number)/
                  molecular_weight_NpO2;
    N_Np237[L]  = N_NpO2[L];
    N_H2O[L]    = (density_H2O_reactor * Avagadros_number)/
                  molecular_weight_H2O;
    N_H[L]      = N_H2O[L];
    N_O[L]      = 2 *(N_UO2[L] + N_PuO2[L] + N_NpO2[L] + N_H2O[L]);
    N_Am241[L] = (density_Am_reactor * Avagadros_number)/ nuclear[17].Aw;
    N_Cm243[L] = (density_Cm_reactor * Avagadros_number)/ nuclear[15].Aw;

   for(k=0; k<NUMBER_OF_ENERGY_GROUPS; k++) {

    Sigma_s[L][k] = ( N_O[L]*nuclear[1].sigma_s[k] +
                      N_U235[L]*nuclear[2].sigma_s[k]+
                      N_U238[L] *nuclear[3].sigma_s[k]+
                      N_Pu239[L] *nuclear[4].sigma_s[k]+
                      N_Np237[L]*nuclear[14].sigma_s[k] +
                      N_H[L]*nuclear[0].sigma_s[k]+
                      N_Am241[L]*nuclear[17].sigma_s[k] +
                      N_Cm243[L]*nuclear[15].sigma_s[k]);

    Sigma_a[L][k] = ( N_O[L]*nuclear[1].sigma_a[k]  +
```

```
                        N_U235[L]*nuclear[2].sigma_a[k]+
                        N_U238[L] *nuclear[3].sigma_a[k]+
                        N_Pu239[L] *nuclear[4].sigma_a[k]+
                        N_Np237[L]*nuclear[14].sigma_a[k] +
                        N_H[L]*nuclear[0].sigma_a[k]+
                        N_Am241[L]*nuclear[17].sigma_a[k] +
                        N_Cm243[L]*nuclear[15].sigma_a[k]);

    Sigma_f[L][k] = ( N_O[L]*nuclear[1].sigma_f[k] +
                        N_U235[L]*nuclear[2].sigma_f[k]+
                        N_U238[L] *nuclear[3].sigma_f[k]+
                        N_Pu239[L] *nuclear[4].sigma_f[k]+
                        N_Np237[L]*nuclear[14].sigma_f[k]+
                        N_H[L]*nuclear[0].sigma_f[k]+
                        N_Am241[L]*nuclear[17].sigma_f[k]+
                        N_Cm243[L]*nuclear[15].sigma_f[k]);

    Sigma_tr[L][k]= (N_U235[L] * nuclear[2].sigma_s[k] *
                       (1-(2.00/(3.00*nuclear[2].A)))) +
                       (N_U238[L] * nuclear[3].sigma_s[k] *
                       (1-(2.00/(3.00*nuclear[3].A)))) +
                       (N_O[L]    * nuclear[1].sigma_s[k] *
                       (1-(2.00/(3.00*nuclear[1].A)))) +
                       (N_Pu239[L]* nuclear[4].sigma_s[k] *
                       (1-(2.00/(3.00*nuclear[4].A)))) +
                       (N_Np237[L]* nuclear[14].sigma_s[k]*
                       (1-(2.00/(3.00*nuclear[14].A))))+
                       (N_H[L]    * nuclear[0].sigma_s[k] *
                       (1-(2.00/(3.00*nuclear[0].A)))) +
                       (N_Am241[L]* nuclear[17].sigma_s[k]*
                       (1-(2.00/(3.00*nuclear[17].A))))+
                       (N_Cm243[L]* nuclear[15].sigma_s[k]*
                        (1- (2.00/(3.00*nuclear[15].A))));

    D[L][k] = 1.00/(3.00*Sigma_tr[L][k]);

    diffusion_length[L][k] = sqrt(D[L][k]/Sigma_a[L][k]);

    Sum_Beta[L][k][2]  = total_delayed_yield[2]/nuclear[2].nu[k];
    Sum_Beta[L][k][3]  = total_delayed_yield[3]/nuclear[3].nu[k];
    Sum_Beta[L][k][4]  = total_delayed_yield[4]/nuclear[4].nu[k];
    Sum_Beta[L][k][14] = total_delayed_yield[14]/nuclear[14].nu[k];
    Sum_Beta[L][k][17] = total_delayed_yield[17]/nuclear[17].nu[k];
    Sum_Beta[L][k][15] = total_delayed_yield[15]/nuclear[15].nu[k];

fission_source[L]  = fission_source[L]+
                     ((1-Sum_Beta[L][k][2])*nuclear[2].nu[k]*
                     N_U235[L]*nuclear[2].sigma_f[k]+(1-Sum_Beta[L][k][3])*
                     nuclear[3].nu[k]*N_U238[L]*nuclear[3].sigma_f[k]+
                     (1-Sum_Beta[L][k][4])*nuclear[4].nu[k]*
                     N_Pu239[L]*nuclear[4].sigma_f[k]+
                     (1-Sum_Beta[L][k][14])*nuclear[14].nu[k]*
                     N_Np237[L]*nuclear[14].sigma_f[k]+
                    (1-Sum_Beta[L][k][15])*nuclear[15].nu[k]*
                    N_Cm243[L]*nuclear[15].sigma_f[k]+
                    (1-Sum_Beta[L][k][17])*nuclear[17].nu[k]*
```

```
                N_Am241[L]*nuclear[17].sigma_f[k]);

        }

    /* Calculating the Group Transfer Cross Sections inside the Core */

        Sigma_s1to0[L] = (N_O[L]*nuclear[1].sigma_s1to0  +
                          N_U235[L]*nuclear[2].sigma_s1to0+
                          N_U238[L]*nuclear[3].sigma_s1to0 +
                          N_Pu239[L] *nuclear[4].sigma_s1to0+
                          N_Np237[L] *nuclear[14].sigma_s1to0+
                          N_H[L]*nuclear[0].sigma_s1to0+
                          N_Am241[L] *nuclear[17].sigma_s1to0+
                          N_Cm243[L] *nuclear[15].sigma_s1to0);

        Sigma_s2to1[L] = (N_O[L]*nuclear[1].sigma_s2to1+
                          N_U235[L]*nuclear[2].sigma_s2to1+
                          N_U238[L]*nuclear[3].sigma_s2to1+
                          N_Pu239[L] *nuclear[4].sigma_s2to1+
                          N_Np237[L] *nuclear[14].sigma_s2to1+
                          N_H[L]*nuclear[0].sigma_s2to1+
                          N_Am241[L] *nuclear[17].sigma_s2to1+
                          N_Cm243[L] *nuclear[15].sigma_s2to1);
      }
      L = L+1;
    }

}


/*************************************************************
*
* B.5.C Calculating the Delayed Neutron Variables in Each Cell
*      for at all Energy Groups (Pu239 Delayed Neutron Groups are Considered)
*
*************************************************************/

   for (L=0; L<N; ++L)  {
    for(i=0; i<6; ++i)   {
     for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k)  {
      beta[L][i][k] = Sum_Beta[L][k][3] * Beta[i];
      delayed_source[L][i][k] = beta[L][i][k] *
                          (nuclear[2].nu[k]*N_U235[L]*nuclear[2].sigma_f[k]
                          + nuclear[3].nu[k]*N_U238[L]*nuclear[3].sigma_f[k]
                          + nuclear[4].nu[k]*N_Pu239[L]*nuclear[4].sigma_f[k]
                          + nuclear[14].nu[k]*N_Np237[L]*nuclear[14].sigma_f[k]
                          + nuclear[17].nu[k]*N_Am241[L]*nuclear[17].sigma_f[k]
                          + nuclear[15].nu[k]*N_Cm243[L]*nuclear[15].sigma_f[k]);
    }
    }
   }


/* Initializing Total Dealyed Sorurce */
  for (L=0; L<N; ++L)  {
   for(i=0; i<6; ++i)   {
```

```
        total_dealyed_sorce[L][i] = 0;
       }
      }

     for(L=0; L<N; ++L) {
      for(i=0; i<6; ++i) {
       for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k) {
        total_dealyed_sorce[L][i] += delayed_source[L][i][k];
       }
      }
     }


 /***********************************************
 *
 * B.6 Initializing Neutron Flux and Coefficient Matrix
 *
 ***********************************************/

   /* Initialize the Nuetron Flux To Zero throught the Reactor */

      for(L=0; L<N; ++L) {
        phi[L] =0.0;
      }

      for(L=0; L<N; ++L) {
        for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k) {
         flux[L][k] =0.0; /* Energy Dependant Flux */
        }
      }

   /* Initialize The Coefficient Matrix To Zero. */

      for(i=0; i<N; i++) {
       for(j=0; j<N; j++) {
        A[i][j] = 0;
       }
      }


/************************************
*
*  B.7 Iteratively Run the
*      Simulation Over Time.
*
************************************/

    start = time(NULL); /* Setting Clock To Find The Time Of Iterations*/

    for(t = 0.0; t < maximum_time; t += time_step) {

  /*********************************
  *
  * B.7.A Fuel Depletion Calculations
  *
  *********************************/
```

```
    for(L=0; L<N; ++L) {
     for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k)  {

      N_U235[L] = N_U235[L] - (N_U235[L] * nuclear[2].sigma_a[k]*
                  flux[L][k])* time_step;
      N_U238[L] = N_U238[L] - (N_U238[L] * nuclear[3].sigma_a[k]*
                  flux[L][k])* time_step;
      N_U239[L] = N_U239[L] + (N_U238[L] * nuclear[3].sigma_c[k] *
                  flux[L][k]- N_U239[L] * decay_const_U239)*time_step;
      N_Np239[L]= N_Np239[L] + (N_U239[L] * decay_const_U239-
                   N_Np239[L] * 0.000003)*time_step;
      N_Pu239[L]= N_Pu239[L] + (N_Np239[L] * 0.000003- N_Pu239[L]*
                   nuclear[4].sigma_a[k]*flux[L][k])*time_step;
     }
    }


 /*****************************************
 *
 * B.7.B Printing the Results to Output Files
 *
 *****************************************/

 /***********************************************
 *
 * B.7.B.a Printing the Distributions in the Reactor
 *
 ************************************************/


  if( write2 <= 20) {

     write2 += 1;

/*Printing Plutonium Distribution */
      fp9 = fopen("Pu.out", "aw");

       for(i=0,L=0; i<Row; ++i) {
        for(j=0; j<Col; ++j) {
         fprintf(fp9, "%lf ",N_Pu239[L]);
         L += 1;
        }
        fprintf(fp9,"\n");
       }
       fprintf(fp9, "\n\n");

      fclose(fp9);

/*Printing Uranium Distribution */
      fp11 = fopen("U.out", "aw");

      for(i=0,L=0; i<Row; ++i) {
       for(j=0; j<Col; ++j) {
        fprintf(fp11, "%lf ",N_U235[L]);
        L += 1;
```

```
        }
        fprintf(fp11,"\n");
       }
        fprintf(fp11, "\n\n");

       fclose(fp11);

 /*Printing Xenon Distributions */

       fp6 = fopen("Xe.out", "aw");

      for(i=0,L=0; i<Row; ++i) {
       for(j=0; j<Col; ++j) {
        fprintf(fp6, "%lf ",N_Xe[L]);
        L += 1;
       }
       fprintf(fp6,"\n");
      }
       fprintf(fp6, "\n\n");

       fclose(fp6);

/*Printing Samarium Distributions */

       fp7 = fopen("Sm.out", "aw");

      for(i=0,L=0; i<Row; ++i) {
       for(j=0; j<Col; ++j) {
        fprintf(fp7, "%lf ",N_Sm[L]);
        L += 1;
       }
       fprintf(fp7,"\n");
      }
       fprintf(fp7, "\n\n");

       fclose(fp7);

  }

  /*********************************************************
  *
  * B.7.B.b Printing the Variations Over Time in the Reactor
  *
  *********************************************************/


  if(((t - time_last_writen1) >= (2*accelerator_period)) || (t == 0.0))  {

    time_last_writen1 = t;

/* Print out of Pu239 Number Density */
       fp8 = fopen("PuNode.out", "aw");
       fprintf(fp8, "%lf %lf\n", t,N_Pu239[60]);
       fclose(fp8);

/* Print out of U235 Number Density */
```

```
      fp10 = fopen("UNode.out", "aw");
      fprintf(fp10, "%lf %lf\n", t,N_U235[60]);
      fclose(fp10);

/* Print out the Xe Flucutations Over Period of Time */

      fp4 = fopen("XeNode.out", "aw");
      fprintf(fp4, "%lf %lf\n", t,N_Xe[60]);
      fclose(fp4);

/*Print out the Samarium Flucutations Over Period of Time */

      fp5 = fopen("SmNode.out", "aw");
      fprintf(fp5, "%lf %lf\n", t,N_Sm[60]);
      fclose(fp5);
  }



/*********************************************************************
*
* B.7.C Calculating the Absorbption Cross-Sections with the Time
*      Dependent Variations of Poisions
*     (Poisons are Considered only in Thermal Group)
*     (Poisions Cross Sections are Available only for Thermal Group)
*
*********************************************************************/

   for (L=0; L<N; ++L) {
     for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k) {
        Sigma_a[L][k] = Sigma_a[L][k] + N_Sb[L]*nuclear[10].sigma_a[k]
                  +N_Te[L]*nuclear[9].sigma_a[k]+ N_I[L]*nuclear[8].sigma_a[k]
                  +N_Xem[L]*nuclear[7].sigma_a[k] +N_Xe[L]*nuclear[7].sigma_a[k]
                  +N_Nd[L]*nuclear[13].sigma_a[k]
                  +N_Pm[L]*nuclear[12].sigma_a[k]
                  +N_Sm[L]*nuclear[11].sigma_a[k];
     }
   }


/*************************************************************
*
* B.7.D Calculating the Group Neutron Removal Cross-Sections
*
*************************************************************/

   for (L=0; L<N; ++L) {
    for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k) {
     if(k==0) {
       Sigma_r[L][k] = Sigma_a[L][k];
     }else if(k==1) {
       Sigma_r[L][k] = Sigma_a[L][k] + Sigma_s1to0[L];
     }else if(k==2) {
       Sigma_r[L][k] = Sigma_a[L][k] + Sigma_s2to1[L];
      }
   }
```

```
  }

/*********************************************************************
*
* B.7.E Calculating the Inscattering Neutrons for Each Energy Group
*
*********************************************************************/

  for (L=0; L<N; ++L)  {
   for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k) {
     if(k==0) {
       In_Scatter_neutrons[L][k] = Sigma_s1to0[L] * flux[L][k+1];
     }else if(k==1) {
       In_Scatter_neutrons[L][k] = Sigma_s2to1[L] * flux[L][k+1];
      }else if(k==2) {
       In_Scatter_neutrons[i][k] = 0;
       }
   }
  }

/*********************************************************************
*
* B.7.F In every Time Step the Delayed Neutrons are Initialized to Zero,
*      which will Avoid the Problems when Adding Delayed Neutrons from
*      All Precursor Groups through a Loop
*
*********************************************************************/

  for (L=0; L<N; ++L) {
    delayed_neutron[L] = 0;
  }

/*********************************************************************
*
* B.7.G Obtaining the Accelerator Source Term in Every Time Step
*      Through a Subroutine (Source Term will be Zero when the Accelerator
*      is not Firing)
*
*********************************************************************/

      source(t);


/*********************************************************************
*
* B.7.H Calculating the Precursor Number densities in each Time Step
*      (Out Side the Energy Group Loop)
*
*********************************************************************/
   for(L=0; L<N; ++L) {
     for(i=0; i<6; ++i) {
       C[L][i] = C[L][i] + (total_dealyed_sorce[L][i] * Total_flux[L]
                   - decay_const_C[i] * C[L][i]) * time_step;
     }

   }
```

```
/*************************************************************
*
* B.7.I Calculating the Coefficient Matrix in each Time Step
*      with New Vaiables (Cross-Sections,Diffustion coeff.)
*
*************************************************************/


   /*  Move Through Each Cell to Form the Coefficient Matrix.  */

 for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k) {

    for(i=0,L=0; i<Row; ++i) {
      for(j=0; j<Col; ++j) {

/* TL node */
       if((i==0) && (j==0))  {

         flux[L][k]  = flux[L+Col][k]/(1+(dz/(2*D[L][k])));

         A[L][L]     = -(1.00/(speed[k]*time_step)) - (1.00*D[L][k]/(dr*dr))
                             -    (2.00*D[L][k]/(dz*dz))
                             -    (D[L][k]/(2*r[L]*dr)) + (X[k]*fission_source[L])
                       - Sigma_r[L][k];
         A[L][L+1]   = D[L+1][k] * ((1.00/pow(dr,2)) + (1.00/(2*r[L]*dr)));
         A[L][L+Col] = D[L+Col][k]/pow(dz,2);
       }

/* TF nodes */

      if((i==0) && (j>0) && (j<Col-1)) {

         flux[L][k] = flux[L+Col][k]/(1+(dz/(2*D[L][k])));

         A[L][L-1]   = D[L-1][k] * ((1.00/pow(dr,2))-(1.00/(2*r[L]*dr)));
         A[L][L]     = -(1.00/(speed[k]*time_step)) - (2.00*D[L][k]/(dr*dr)) -
                       (2.00*D[L][k]/(dz*dz))+
                        (X[k]* fission_source[L]) - Sigma_r[L][k];
         A[L][L+1]   = D[L+1][k] * ((1.00/pow(dr,2))+(1.00/(2*r[L]*dr)));
         A[L][L+Col] = D[L+Col][k]/pow(dz,2);
       }

/* TR node */
      if((i==0) && (j==Col-1)) {

         flux[L][k] = flux[L+Col][k]/(1+(dz/(2*D[L][k])));
         temp   = flux[L][k];
         flux[L][k] = flux[L-1][k]/(1+(dr/(2*D[L][k])));
         flux[L][k] = (flux[L][k] + temp)/2.00;

         A[L][L-1]   = D[L-1][k] * ((1.00/pow(dr,2))-(1.00/(2*r[L]*dr)));
         A[L][L]     = -(1.00/(speed[k]*time_step)) - (2.00*D[L][k]/(dr*dr)) -
                       (2.00*D[L][k]/(dz*dz))+ (X[k]*fission_source[L]) -
                       Sigma_r[L][k];
```

```
        A[L][L+Col] = D[L+Col][k]/pow(dz,2);
      }

/* LF Nodes */
      if((i>0) && (i<Row-1) && (j==0)) {
        A[L][L-Col] = D[L-Col][k]/pow(dz,2);
        A[L][L]    = -(1.00/(speed[k]*time_step))- (1.00*D[L][k]/(dr*dr)) -
                     (2.00*D[L][k]/(dz*dz))(D[L][k]/(2*r[L]*dr)) +
                          (X[k]*fission_source[L])- Sigma_r[L][k];
        A[L][L+1]  = D[L+1][k] * ((1.00/pow(dr,2))+ (1.00/(2*r[L]*dr)));
        A[L][L+Col] = D[L+Col][k]/pow(dz,2);
      }

/* Central Nodes */
      if((i>0) && (i<Row-1) && (j>0) && (j<Col-1)) {
        A[L][L-Col] = D[L-Col][k]/pow(dz,2);
        A[L][L-1]  = D[L-1][k] * ((1.00/pow(dr,2))-(1.00/(2*r[L]*dr)));
        A[L][L]    = -(1.00/(speed[k]*time_step))- (2.00*D[L][k]/(dr*dr)) -
                     (2.00*D[L][k]/(dz*dz))+ (X[k]*fission_source[L]) -
                      Sigma_r[L][k];
        A[L][L+1]  = D[L+1][k] * ((1.00/pow(dr,2))+(1.00/(2*r[L]*dr)));
        A[L][L+Col] = D[L+Col][k]/pow(dz,2);
      }

/* RF Nodes */
      if((i>0) && (i<Row-1) && (j>0) && (j==Col-1)) {

        flux[L][k] = flux[L-1][k]/(1+(dr/(2*D[L][k])));

        A[L][L-Col] = D[L-Col][k]/pow(dz,2);
        A[L][L-1]  = D[L-1][k] * ((1.00/pow(dr,2))-(1.00/(2*r[L]*dr)));
        A[L][L]    = -(1.00/(speed[k]*time_step)) - (2.00*D[L][k]/(dr*dr)) -
                     (2.00*D[L][k]/(dz*dz))+ (X[k]*fission_source[L]) -
                     Sigma_r[L][k];
        A[L][L+Col] = D[L-Col][k]/pow(dz,2);
      }

/* BL Node */
      if((i==Row-1) && (j==0))  {

        flux[L][k] = flux[L-Col][k]/(1+(dz/(2*D[L][k])));

        A[L][L-Col] = D[L-Col][k]/pow(dz,2);
        A[L][L]    = -(1.00/(speed[k]*time_step)) - (1.00*D[L][k]/(dr*dr)) -
                     (2.00*D[L][k]/(dz*dz)) - (D[L][k]/(2*r[L]*dr)) +
                     (X[k]*fission_source[L])- Sigma_r[L][k];
        A[L][L+1]  = D[L+1][k] * ((1.00/pow(dr,2))+(1.00/(2*r[L]*dr)));
      }

/* BF Nodes */
      if((i==Row-1) && (j>0) && (j<Col-1))  {

        flux[L][k] = flux[L-Col][k]/(1+(dz/(2*D[L][k])));

        A[L][L-Col] = D[L-Col][k]/pow(dz,2);
        A[L][L-1]  = (D[L-1][k] * ((1.00/pow(dr,2))) - (1.00/(2*r[L]*dr)));
```

```
          A[L][L]    = -(1.00/(speed[k]*time_step))- (2.00*D[L][k]/(dr*dr)) -
                       (2.00*D[L][k]/(dz*dz))+ (X[k]*fission_source[L]) -
                       Sigma_r[L][k];
          A[L][L+1]  = D[L+1][k] * ((1.00/pow(dr,2))+(1.00/(2*r[L]*dr)));

        }

  /* BR Node */
        if((i==Row-1) && (j==Col-1)) {

          flux[L][k] = flux[L-Col][k]/(1+(dz/(2*D[L][k])));
          temp  = flux[L][k];
          flux[L][k] = flux[L-1][k]/(1+(dr/(2*D[L][k])));
          flux[L][k] = (flux[L][k] + temp)/2.00;

          A[L][L-Col] = D[L-Col][k]/pow(dz,2);
          A[L][L-1]  = D[L-1][k] * ((1.00/pow(dr,2))-(1.00/(2*r[L]*dr)));
          A[L][L]    = -(1.00/(speed[k]*time_step))- (2.00*D[L][k]/(dr*dr)) -
                       (2.00*D[L][k]/(dz*dz))+ (X[k]*fission_source[L]) -
                       Sigma_r[L][k];
        }
        ++L;
      }
    }

/*****************************************************************************
*
* B.7.J Calculating the Total no. of Neutrons Produced from All Precursor Groups
*
*****************************************************************************/

     for(L=0; L<N; ++L) {
       for(i=0; i<6; ++i) {
         delayed_neutron[L] = delayed_neutron[L] + (decay_const_C[i] * C[L][i]);
       }

     }

/*************************************************************************
*
* B.7.K Calculating the Matrix from the R.H.S Values of Diffustion Equations
*
*************************************************************************/

    for (L=0; L<N; ++L)  {
      if(k==0 || k==1) {
        B[L] =  -(1.00/(speed[k]*time_step))* S[L][k] - (1.00/(speed[k]*time_step))*
                flux[L][k]- f[k]*delayed_neutron[L] - In_Scatter_neutrons[L][k];
      }
     else if(k==2) {
        B[L] =  -(1.00/(speed[k]*time_step))* S[L][k] - (1.00/(speed[k]*time_step))*
                 flux[L][k]- f[k]*delayed_neutron[L] ;
      }

    }
```

```
/*************************************************************************
*
* B.7.L Since Gauss-Seidel Can Handle only 1-D; 2-D Flux is
*      Assigned into 1-D Flux Variable
*
*************************************************************************/

  for (L=0; L<N; ++L)  {
    phi[L] = flux[L][k];
  }

/*************************************************************************
*
* B.7.M Flux in the New Time is Calculated by calling the Subroutine which
*      is in a Separate Header File MATRIX.H file
*
*************************************************************************/

  matrix_gauss_seidel (tolerance, N, A, phi, B);


/*************************************************************************
*
* B.7.N Again 2-D Flux is Assigned to 1-D Flux Variables, after Matrix Operation
*
*************************************************************************/

  for (L=0; L<N; ++L)  {
    flux[L][k] = phi[L];
  }

} /* Closing Energy Group Loop of Phi Calculations */


/*************************************************************
*
* B.7.O Calculating Number Densities of Poisons in Each Time Step
*
*************************************************************/

 for(L=0;L<N; ++L) {
  for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k) {

   N_Sb[L] = (fission_yield_Sb * Sigma_f[L][k] * flux[L][k] -
              decay_const_Sb * N_Sb[L]) * time_step + N_Sb[L];

   N_Te[L] = (fission_yield_Te * Sigma_f[L][k] * flux[L][k] +
              decay_const_Sb * N_Sb[L]- decay_const_Te * N_Te[L])*
              time_step + N_Te[L];

   N_I[L]  = (fission_yield_I * Sigma_f[L][k] * flux[L][k] +
             decay_const_Te * N_Te[L]- decay_const_I * N_I[L]-
             nuclear[8].sigma_a[k] * flux[L][k] * N_I[L]) * time_step + N_I[L];

   N_Xem[L] = (fission_yield_Xem * Sigma_f[L][k] * flux[L][k] +
               0.09 * decay_const_I * N_I[L]- decay_const_Xem * N_Xem[L] -
```

```
                nuclear[7].sigma_a[k] * flux[L][k] * N_Xem[L]) * time_step +
                N_Xem[L];

    N_Xe[L] = (fission_yield_Xe * Sigma_f[L][k] * flux[L][k] +
              0.91 * decay_const_I * N_I[L]+ decay_const_Xem * N_Xem[L] -
              decay_const_Xe * N_Xe[L] - nuclear[7].sigma_a[k]*
              flux[L][k] * N_Xe[L]) * time_step + N_Xe[L];

    N_Nd[L] = (fission_yield_Nd * Sigma_f[L][k] * flux[L][k] - decay_const_Nd *
              N_Nd[L]) * time_step + N_Nd[L];

    N_Pm[L] = (decay_const_Nd * N_Nd[L] - decay_const_Pm * N_Pm[L]) *
               time_step + N_Pm[L];

    N_Sm[L] = (decay_const_Pm * N_Pm[L] - nuclear[11].sigma_a[k] * flux[L][k] *
               N_Sm[L]) * time_step + N_Sm[L];

   }

  }

/*****************************************************************************
*
* B.7.P Calculating the Total flux from All the Energy Groups in Each Time Step
*
*****************************************************************************/

  for(L=0;L<N; ++L) {       /* Initializing the Total Flux in Each Time Step */
    Total_flux[L] = 0.0;    /* Before Calculating the Latest Flux            */
    Total_S[L]  = 0.0;
  }

  for(L=0;L<N; ++L) {
   for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k) {
     Total_flux[L] = Total_flux[L] + flux[L][k];
     Total_S[L]    = Total_S[L] + S[L][k];
   }
  }


/*************************************************************
*
* B.7.Q Print Out the Results (Distributions) to Out Put Files
*
*************************************************************/

 if( write <= 20) {

   write += 1;

/* Print out the Source Flux for a Contour or Surface Plot */
  fp12 = fopen("sourceflux.out", "aw");

  for(i=0,L=0; i<Row; ++i) {
    for(j=0; j<Col; ++j) {
      fprintf(fp12, "%lf ",Total_S[L]);
```

```
        L += 1;
       }
        fprintf(fp12,"\n");
    }
        fprintf(fp12, "\n\n\n");
        fclose(fp12);


  /* Print out the Neutron Flux Distribution */

    fp2 = fopen("flux.out", "aw");

    for(i=0,L=0; i<Row; ++i) {
      for(j=0; j<Col; ++j) {
        fprintf(fp2, "%lf ",Total_flux[L]);
        L += 1;
      }
        fprintf(fp2,"\n");

    }
        fprintf(fp2, "\n\n\n");
        fclose(fp2);


   }

/*****************************************************************
*
* B.7.R Printing the Results (Time Dependent) to Output Files
*
*****************************************************************/

  if(((t - time_last_writen) >= (2*accelerator_period)) || (t == 0.0))  {

    time_last_writen = t;

  /* Print the Flux Variation at a Node */

        fp3 = fopen("nodeflux.out", "aw");
        fprintf(fp3, "%lf  %lf\n", t,Total_flux[60]);
        fclose(fp3);
  }

  /* Print out the Node Flux When the Accelerator is not Firing */

 if(((t - time_last_writen2) >= (2*accelerator_period)) || (t == time_step))  {

   time_last_writen2 = t;

          fp13 = fopen("nodeflux2.out", "aw");
          fprintf(fp13, "%lf  %lf\n", t,Total_flux[60]);
          fclose(fp13);
        }

 }  /* Closing Time step loop */
```

```
    end = time(NULL);
 printf("Time Started = %d\n", start);
 printf("Time ended = %d\n", end);
    elapsed = (double) (end - start); /* in seconds */
    elapsed = elapsed/3600;   /* in hours */
 printf("Time elaplsed = %lf hours\n", elapsed);


  /*********************************************
  *
  *  C. Print Out Summary of Simulation to a File
  *
  *********************************************/

  fp_results = fopen("Results.out", "aw");

fprintf(fp_results,"\n\nTHE SIMULATION RESULTS ALONG WITH THE IMPORTANT INPUT
SPECIFICATIONS ARE SUMMARISED IN THE FILE\n\n");

fprintf(fp_results,"The acclerator coupled reactor was analysed for the transient
        behavior.\n");
fprintf(fp_results,"Time step of the simulations    : %lf sec\n\n", time_step);

fprintf(fp_results,"************************************\n");
fprintf(fp_results,"*  Program Input Details       *\n");
fprintf(fp_results,"************************************\n");
fprintf(fp_results,"\n\n");
fprintf(fp_results,"A. Reactor Description          \n\n");
fprintf(fp_results,"As a Test Case:The subcritical blanket is filed with uranium,
        plutonium oxides and some minar actinides.\n");
fprintf(fp_results,"The volumetric percent of core and water in the reactor are %lf and
        %lf respectively.\n",percent_core,percent_H2O);
fprintf(fp_results,"The percent by weight of Cm, Am, UO2, PuO2, NpO2 are %lf %lf %lf
        %lf %lf respectively.\n", percent_Cm_core, percent_Am_core,percent_UO2_core,
        percent_PuO2_core, percent_NpO2_core);
fprintf(fp_results,"The reactor is shileded by steel reflector (Fe (%lf),  Cr(%lf)).\n\n",
        percent_weight_Fe,percent_weight_Cr);

fprintf(fp_results,"In this analysis neutron transport/diffusion equations were used which
        are solved");
fprintf(fp_results," by finite volume method.\n");
fprintf(fp_results,"   Temperature (K)                 : %lf\n",reactor_temperature);
fprintf(fp_results,"   Radius of the reactor           : %d cm\n", reactor_dimension_R);
fprintf(fp_results,"   Finite voumes in the radial direction : %d\n", reactor_R_cells);
fprintf(fp_results,"   Axial dimention of the reactor      : %d cm\n",
        reactor_dimension_Z);
fprintf(fp_results,"   Finite voumes in the axial direction  : %d\n\n", reactor_Z_cells);
fprintf(fp_results,"Three neutron energy groups (thermal = %lf ev, epithermal=%lf ev,
        fast=%lfev) were assumed.\n\n", Ave_energy_group[0], Ave_energy_group[1],
        Ave_energy_group[2]);
fprintf(fp_results,"All the fission and delaed neutrons were assumed to be produced in the
        fast group.\n\n");
fprintf(fp_results,"All the neutron interaction cross sections for this analysis were
        obtained from ENDFB libraries of MCNPX.\n\n");

fprintf(fp_results,"B.  Accelerator Description          \n");
```

```
fprintf(fp_results,"   1. Accelerator Particle      :%d
      (3=electron;9=proton)\n",accelerator_particle);
fprintf(fp_results,"   2. Particle Energy (MeV)     :%lf\n",particle_energy);
fprintf(fp_results,"   3. Accelerator Current (A)   :%lf\n",accelerator_current);
fprintf(fp_results,"   4. Acce. Frequency (Hz)      :%lf\n",accelerator_frequency);
fprintf(fp_results,"   5. Acc. Duty Cycles (micros) :%lf\n\n\n",accelerator_duty_cycle);

fprintf(fp_results,"************************************\n");
fprintf(fp_results,"*   Program Output                 *\n");
fprintf(fp_results,"************************************\n");
fprintf(fp_results,"\n\n");

/* Printing the Message About the Reactor Multiplication */
fprintf(fp_results,"The blanket was analysed for criticality using MCNPX \n");
          if(keff > 1) {

            fprintf(fp_results," Keff = %lf  std_keff = %lf  \n", keff,std_keff);
            fprintf(fp_results,"Reactor is super critical: consdier changing the geometry or
                  the material composition of the reactor \n");

          } else if (keff == 1) {

            fprintf(fp_results," Keff = %lf  std_keff = %lf   \n", keff,std_keff);
            fprintf(fp_results,"Reactor is critical: consdier changing the geometry or the
                  material composition of the reactor\n");

           } else {

             fprintf(fp_results," Keff = %lf  std_keff = %lf, Reactor is sub critical\n\n",
                  keff,std_keff);
           }

    fprintf(fp_results,"MCNPX simulations were done to find the neutron production in
          the target");
    fprintf(fp_results,"and the subcritical blanket.\n\n");
    fprintf(fp_results,"Speed of the neutrons in each group are:\n");

      for(k=0; k<NUMBER_OF_ENERGY_GROUPS; k++) {
    fprintf(fp_results,"     speed[%d] : %lf\n cm/sec", k, speed[k]);
      }

    fprintf(fp_results,"The diffusion coefficient and diffusion length in the reactor and
          reflector are:\n");
    fprintf(fp_results,"  Diffusion coefficient in the core :\n");
    fprintf(fp_results,"          Thermal   : %lf cm\n",D[60][0]);
    fprintf(fp_results,"          Epihermal : %lf cm\n",D[60][1]);
    fprintf(fp_results,"          Fast      : %lf cm\n",D[60][2]);
    fprintf(fp_results,"  Diffusion coefficient in the reflector :\n");
    fprintf(fp_results,"          Thermal   : %lf cm\n",D[0][0]);
    fprintf(fp_results,"          Epihermal : %lf cm\n",D[0][1]);
    fprintf(fp_results,"          Fast      : %lf cm\n",D[0][2]);
    fprintf(fp_results,"  Diffusion lenght in the core :\n");
    fprintf(fp_results,"          Thermal   : %lf cm\n",diffusion_length[60][0]);
    fprintf(fp_results,"          Epihermal : %lf cm\n",diffusion_length[60][1]);
    fprintf(fp_results,"          Fast      : %lf cm\n",diffusion_length[60][2]);
    fprintf(fp_results,"  Diffusion length in the reflector :\n");
```

```
    fprintf(fp_results,"          Thermal   : %lf cm\n",diffusion_length[0][0]);
    fprintf(fp_results,"          Epihermal : %lf cm\n",diffusion_length[0][1]);
    fprintf(fp_results,"          Fast      : %lf cm\n",diffusion_length[0][2]);

    fprintf(fp_results,"Time elaplsed for the %lf sec of reactor simulation = %lf
          hours\n\n",maximum_time,elapsed);

  fclose(fp_results);
  /***********************************
  *
  *  D. Terminate the Program.
  *
  ***********************************/

    fclose;

}


/*****************************************************
*
* Subroutine "get_properties"
* (To Get Cross-Section and other Info. of Isotopes
*
*****************************************************/

int get_properties() {
    char input_line[80];
    FILE *fp;
    int i,j;
    fp = fopen("nuclear.dat","rt");
     for(i=0; feof(fp) == 0; ++i) {
        fscanf(fp, "%d", &nuclear[i].Z);
        fscanf(fp, "%d %d %s %s", &nuclear[i].A, &nuclear[i].ID, nuclear[i].name,
              nuclear[i].isotope);
        fscanf(fp, "%lf %lf", &nuclear[i].Aw, &nuclear[i].density);

  /*Reading Cross-Section Data at Different Energy Groups*/
         for(j=0;j<NUMBER_OF_ENERGY_GROUPS;j++){
           fscanf(fp, "%lf ", &nuclear[i].sigma_c[j]);
           fscanf(fp, "%lf ", &nuclear[i].sigma_s[j]);
           fscanf(fp, "%lf ", &nuclear[i].sigma_f[j]);
           fscanf(fp, "%lf ", &nuclear[i].nu[j]);
           printf("Sigma Capture   : %lf\n", nuclear[i].sigma_c[j]);
           printf("Sigma Scattering  : %lf\n", nuclear[i].sigma_s[j]);
           printf("Sigma Fission   : %lf\n", nuclear[i].sigma_f[j]);
           printf("Nu   : %lf\n", nuclear[i].nu[j]);

    /* Converting the Cross-Sections from Barns to cm2 */

        nuclear[i].sigma_c[j] = nuclear[i].sigma_c[j] * pow(10,-24);
        nuclear[i].sigma_s[j] = nuclear[i].sigma_s[j] * pow(10,-24);
        nuclear[i].sigma_f[j] = nuclear[i].sigma_f[j] * pow(10,-24);
          }

   /* Reading the Group Transfer Cross-Sections */
```

```
            fscanf(fp, "%lf ", &nuclear[i].sigma_s1to0);
            fscanf(fp, "%lf ", &nuclear[i].sigma_s2to1);

          nuclear[i].sigma_s1to0 = nuclear[i].sigma_s1to0 * pow(10,-24);
          nuclear[i].sigma_s2to1 = nuclear[i].sigma_s2to1 * pow(10,-24);
         }

    fclose(fp);
}


/*************************************************************************
*
*  Subroutine:  get_setupdata()
*  Purpose:     Input User Specified Data from a File called:  SETUP.DAT
*
*************************************************************************/

int get_setupdata()  {

   FILE *fp;

   fp = fopen("setup.dat", "r");

      fscanf(fp,"%lf", &reactor_temperature);
      fscanf(fp,"%d",  &accelerator_particle);
      fscanf(fp,"%lf", &particle_energy);
      fscanf(fp,"%lf", &accelerator_current);
      fscanf(fp,"%lf", &accelerator_pulse_charge);
      fscanf(fp,"%lf", &accelerator_frequency);
      accelerator_period = 1.0 / accelerator_frequency;
      fscanf(fp,"%lf", &accelerator_duty_cycle);
      fscanf(fp,"%d",  &target_material);

   fclose(fp);
}


/*************************************************************************************
*
*  Subroutine:  cell_dimensions()
*  Purpose:     To Divide the Reactor into Finite Cells and Get their Coordinate Info.
*
*************************************************************************************
*/

double cell_dimensions()  {

  dr =(double)(reactor_dimension_R-thickness_shield)/reactor_R_cells;
  dz =(double)(reactor_dimension_Z-thickness_shield)/reactor_Z_cells;

  count = 0;

  for (i=0; i<reactor_Z_cells; i++)  {
    for(j=0; j<reactor_R_cells; j++)  {
```

```
        cell[i][j].r_min = j*dr;
        cell[i][j].r_max = cell[i][j].r_min + dr;
        cell[i][j].z_min = i*dz;
        cell[i][j].z_max = cell[i][j].z_min + dz;
       count = count + 1;

    }
  }

}


/************************************************************
*
*  Subroutine:  make_mcnpx_keff()
*  Purpose:    To Make a MCNPX Input File to Find keff
*
************************************************************/

int make_mcnpx_keff()  {
 int cell_number;

 double center_r;
 double center_z;

 FILE *fp_mcnpx1;

 fp_mcnpx1 = fopen("C:\\mcnpx2\\BIN\\Keff","w");

        fprintf(fp_mcnpx1, "Trial\n");
        fprintf(fp_mcnpx1, "c cell cards\n");

            fprintf(fp_mcnpx1, "1 0  -1  imp:e,p,h,n 1    $Vaccume Beam Pipe\n");
            fprintf(fp_mcnpx1, "c \n");

    for(i=0,cell_number = 2; i<reactor_Z_cells; ++i) {
      for(j=0; j<reactor_R_cells; ++j) {

        if(i<3 && j==0){
          fprintf(fp_mcnpx1, "%d 2 -%lf -%d 1  imp:e,p,h,n 1  $Reactor Blanket \n",
                cell_number,density_reactor, cell_number,cell_number-1);
            }
     /* Target */
           else if(i==3 && j==0){
             if(target_material==73){
               fprintf(fp_mcnpx1, "%d 1 -16.65 -%d  imp:e,p,h,n 1  $Target \n",
                      cell_number, cell_number);
             }
             if(target_material==74) {
               fprintf(fp_mcnpx1, "%d 1 -19.3 -%d  imp:e,p,h,n 1  $Target \n",
                       cell_number, cell_number);
             }
             if(target_material==82) {
               fprintf(fp_mcnpx1, "%d 1 -11.34 -%d  imp:e,p,h,n 1  $Target \n",
                       cell_number, cell_number);
             }
```

```
            if(target_material==83) {
              fprintf(fp_mcnpx1, "%d 1 -9.78 -%d  imp:e,p,h,n 1  $Target \n",
                      cell_number, cell_number);
            }
            if(target_material==92) {
              fprintf(fp_mcnpx1, "%d 1 -17.1 -%d  imp:e,p,h,n 1  $Target \n",
                      cell_number, cell_number);
            }
          }
          else if(j==0){
             fprintf(fp_mcnpx1, "%d 2 -%lf -%d    imp:e,p,h,n 1  $Reactor Blanket\n",
                    cell_number,density_reactor, cell_number);
           }

          else {
           fprintf(fp_mcnpx1, "%d 2 -%lf -%d %d  imp:e,p,h,n 1  $Reactor Blanket
                  \n", cell_number,density_reactor, cell_number,cell_number-1);
          }


           ++cell_number;
    }
          fprintf(fp_mcnpx1, "c \n");
          Circum_cell[i] = cell_number-1;

  }

  /* Shielding */
    fprintf(fp_mcnpx1, "%d 3 -7.8 -%d \n", cell_number, cell_number);
    fprintf(fp_mcnpx1,"    ");
    for(i=0; i<reactor_Z_cells; ++i) {
      fprintf(fp_mcnpx1," %d", Circum_cell[i]);
    }
    fprintf(fp_mcnpx1,"\n     imp:e,p,h,n 1  $Steel Shielding\n");

  /* Vaccume Surroundings */
   fprintf(fp_mcnpx1, "%d 0      %d imp:e,p,h,n 0   $Vaccume surroundings\n",
            cell_number+1, cell_number);
   fprintf(fp_mcnpx1, "\n");
   fprintf(fp_mcnpx1,"c Surface cards. \n");
   fprintf(fp_mcnpx1,"c RCC cells \n");

 /* Vaccum Beam Pipe with 0.5 cm Radius*/
   fprintf(fp_mcnpx1, "1 RCC 0 0 0 0 0 0 %lf 0.5 \n",3*dz);
   fprintf(fp_mcnpx1, "c \n");

 cell_number = 2;

  for(i=0; i<reactor_Z_cells; ++i) {
    for(j=0; j<reactor_R_cells; ++j) {
        if(i==3 && j==0){
 /* Target */
   fprintf(fp_mcnpx1,"%d RCC 0 0 %lf 0 0 %lf %lf \n",cell_number,3*dz,dz,dr);
        }
        else{
          if(cell[i][j].z_min!=cell[i-1][j].z_max) {
```

```
          cell[i][j].z_min = cell[i-1][j].z_max;
         }
         fprintf(fp_mcnpx1,"%d RCC 0 0 %lf 0 0 %lf %lf \n",cell_number,
             cell[i][j].z_min,dz, cell[i][j].r_max);
       }
       ++cell_number;

   }
       fprintf(fp_mcnpx1, "c \n");
}


/* Reactor Shielding */
       fprintf(fp_mcnpx1, "%d RCC 0 0 -%lf 0 0 %lf %lf \n", cell_number,
              thickness_shield,reactor_dimension_Z+thickness_shield,
              reactor_dimension_R+thickness_shield);


 fprintf(fp_mcnpx1, "\n");
 fprintf(fp_mcnpx1, "c Data cards\n");
 fprintf(fp_mcnpx1, "mode e p h n\n");

 if(target_material==82) {
 fprintf(fp_mcnpx1, "m1 82206 -24.44 82207 -22.94 82208 -52.647\n");
 }
 if(target_material==8283) {
 fprintf(fp_mcnpx1, "m1 82206 -13.625 82207 -12.515 82208 -29.36 83209 -44.5
        \n");
 }

 if(target_material==74){
 fprintf(fp_mcnpx1, "m1 74182 26.62 74183 14.31 74184 30.64 74186 28.43\n");
 }
  else if(target_material==83){
 fprintf(fp_mcnpx1, "m1 83206 1 \n");
 }
 if(target_material==92){
 fprintf(fp_mcnpx1, "m1 92235 -0.7 92238 -99.3 \n");
 }
 if(target_material==73) {
 fprintf(fp_mcnpx1, "m1 73181 1 \n");
 }


 fprintf(fp_mcnpx1, "m2 92235 -%lf 92238 -%lf 94239 -%lf 93237 -%lf 95241 \n",
        percent_U235_reactor,percent_U238_reactor,percent_Pu_reactor,
        percent_Np_reactor);

 fprintf(fp_mcnpx1,"    -%lf 96243 -%lf 1001 -%lf 8016 -
        %lf\n",percent_Am_reactor,percent_Cm_reactor,
        percent_H_reactor, percent_O2_reactor);

 fprintf(fp_mcnpx1, "m3 26000 -%lf 24000 -%lf\n", percent_weight_Fe,
        percent_weight_Cr);

 fprintf(fp_mcnpx1, "cut:e j 6.7\n");
```

```
        fprintf(fp_mcnpx1, "cut:p j 6.7\n");

       center_z = reactor_dimension_Z/2.00;

        fprintf(fp_mcnpx1, "print\n");
        fprintf(fp_mcnpx1, "kcode 1000 1.0 10 200\n");
        fprintf(fp_mcnpx1, "ksrc 0 0 %lf\n", center_z);

   fclose(fp_mcnpx1);

}

/************************************************************
*
*  Subroutine:  make_mcnpx_keff()
*  Purpose:    To Make a MCNPX Input File to Simulate the
*              Neutron Production from the Target
*
************************************************************/


int make_mcnpx_Neutron1()  {
 int cell_number;
 int skipline;
 double center_r;
 double center_z;

 FILE *fp_mcnpx2;

 fp_mcnpx2 = fopen("C:\\MCNPX2\\BIN\\Neutron1","w");

        fprintf(fp_mcnpx2, "Trial\n");
        fprintf(fp_mcnpx2, "c cell cards\n");

  center_r = reactor_dimension_R/2.00;
  center_z = reactor_dimension_Z/2.00;

            fprintf(fp_mcnpx2, "1 0  -1     ");
             if(accelerator_particle == 9) {
              fprintf(fp_mcnpx2, "imp:h,n,p = 1    $Vaccume Beam Pipe\n");
             } else if(accelerator_particle == 3) {
              fprintf(fp_mcnpx2, "imp:e,p,n = 1    $Vaccume Beam Pipe\n");
              }
            fprintf(fp_mcnpx2, "c \n");


    for(i=0,cell_number = 2; i<reactor_Z_cells; ++i) {
      for(j=0; j<reactor_R_cells; ++j) {

          if(i<center_cell_z && j==0){
           fprintf(fp_mcnpx2, "%d 2 -%lf -%d 1      ", cell_number,density_reactor,
                 cell_number,cell_number-1);
             if(accelerator_particle == 9) {
               fprintf(fp_mcnpx2, "imp:h,n,p=1  $Reactor Blanket \n" );
             } else if(accelerator_particle == 3) {
              fprintf(fp_mcnpx2, "imp:e,p,n=1  $Reactor Blanket \n" );
```

```
            }
        }
/* Target */
        else if(i==center_cell_z && j==0){
          if(target_material==73){

            fprintf(fp_mcnpx2, "%d 1 -16.65 -%d     ",cell_number, cell_number);
              if(accelerator_particle == 9) {
                fprintf(fp_mcnpx2,"imp:h,n,p=1  $Target \n");
              } else if(accelerator_particle == 3) {
                fprintf(fp_mcnpx2,"imp:e,p,n=1  $Target \n");
               }
          }
          if(target_material==74) {

            fprintf(fp_mcnpx2, "%d 1 -19.3 -%d      ", cell_number, cell_number);
              if(accelerator_particle == 9) {
                fprintf(fp_mcnpx2,"imp:h,n,p=1  $Target \n");
              } else if(accelerator_particle == 3) {
                fprintf(fp_mcnpx2,"imp:e,p,n=1  $Target \n");
               }

          }
          if(target_material==82) {

            fprintf(fp_mcnpx2, "%d 1 -11.34 -%d     ", cell_number, cell_number);
              if(accelerator_particle == 9) {
                fprintf(fp_mcnpx2,"imp:h,n,p=1  $Target \n");
              } else if(accelerator_particle == 3) {
                fprintf(fp_mcnpx2,"imp:e,p,n=1  $Target \n");
               }

          }
          if(target_material==83) {

            fprintf(fp_mcnpx2, "%d 1 -9.78 -%d      ", cell_number, cell_number);
              if(accelerator_particle == 9) {
                fprintf(fp_mcnpx2,"imp:h,n,p=1  $Target \n");
              } else if(accelerator_particle == 3) {
                fprintf(fp_mcnpx2,"imp:e,p,n=1  $Target \n");
               }

          }
          if(target_material==92) {

            fprintf(fp_mcnpx2, "%d 1 -17.1 -%d      ", cell_number, cell_number);
              if(accelerator_particle == 9) {
                fprintf(fp_mcnpx2,"imp:h,n,p=1  $Target \n");
              } else if(accelerator_particle == 3) {
                fprintf(fp_mcnpx2,"imp:e,p,n=1  $Target \n");
               }

          }
        }
        else if(j==0){
```

```
fprintf(fp_mcnpx2, "%d 2 -%lf -%d      ", cell_number,density_reactor, cell_number);
            if(accelerator_particle == 9) {
              fprintf(fp_mcnpx2,"imp:h,n,p=1  $Reactor Blanket \n");
                    } else if(accelerator_particle == 3) {
               fprintf(fp_mcnpx2,"imp:e,p,n=1  $Reactor Blanket \n");
                     }

              }

            else {

fprintf(fp_mcnpx2, "%d 2 -%lf -%d %d       ", cell_number,density_reactor,
       cell_number,cell_number-1);
                 if(accelerator_particle == 9) {
                   fprintf(fp_mcnpx2,"imp:h,n,p=1  $Reactor Blanket \n");
                 } else if(accelerator_particle == 3) {
                   fprintf(fp_mcnpx2,"imp:e,p,n=1  $Reactor Blanket \n");
                  }

            }


              ++cell_number;
       }
            fprintf(fp_mcnpx2, "c \n");
            Circum_cell[i] = cell_number-1;

     }

     /* Shielding */

       fprintf(fp_mcnpx2, "%d 3 -7.8 -%d \n", cell_number, cell_number);
       fprintf(fp_mcnpx2,"    ");
         for(i=0; i<reactor_Z_cells; ++i) {
            fprintf(fp_mcnpx2," %d", Circum_cell[i]);
            if(i >= 402) {
             fprintf(fp_mcnpx2,"\n    ");
            }
         }
         if(accelerator_particle == 9) {
           fprintf(fp_mcnpx2,"\n     imp:h,n,p=1  $Steel Shielding\n");
         } else if(accelerator_particle == 3) {
           fprintf(fp_mcnpx2,"\n     imp:e,p,n=1  $Steel Shielding\n");
          }

     /* Vaccume Surroundings */

       fprintf(fp_mcnpx2, "%d 0      %d      ", cell_number+1, cell_number);
         if(accelerator_particle == 9) {
           fprintf(fp_mcnpx2," imp:h,n,p=0  $Vaccume surroundings\n");
         } else if(accelerator_particle == 3) {
           fprintf(fp_mcnpx2," imp:e,p,n=0  $Vaccume surroundings\n");
          }
       fprintf(fp_mcnpx2, "\n");
       fprintf(fp_mcnpx2,"c Surface cards. \n");
       fprintf(fp_mcnpx2,"c RCC cells \n");
```

```
/* Vaccum Beam Pipe with 0.5 cm Radius*/

fprintf(fp_mcnpx2, "1 RCC 0 0 0 0 0 %lf 0.5 \n",center_cell_z*dz);
fprintf(fp_mcnpx2, "c \n");

cell_number = 2;

for(i=0; i<reactor_Z_cells; ++i) {
 for(j=0; j<reactor_R_cells; ++j) {
     if(i==center_cell_z && j==0){

/* Target */

fprintf(fp_mcnpx2,"%d RCC 0 0 %lf 0 0 %lf %lf
        \n",cell_number,center_cell_z*dz,dz,dr);
     }
     else{
      if(cell[i][j].z_min!=cell[i-1][j].z_max) {
       cell[i][j].z_min = cell[i-1][j].z_max;
      }
      fprintf(fp_mcnpx2,"%d RCC 0 0 %lf 0 0 %lf %lf \n",cell_number,
            cell[i][j].z_min,dz, cell[i][j].r_max);
     }
     ++cell_number;

 }
     fprintf(fp_mcnpx2, "c \n");
}

/* Reactor Shielding */

 fprintf(fp_mcnpx2, "%d RCC 0 0 -%lf 0 0 %lf %lf \n", cell_number,
        thickness_shield,reactor_dimension_Z+thickness_shield,
        reactor_dimension_R+thickness_shield);

fprintf(fp_mcnpx2, "\n");
fprintf(fp_mcnpx2, "c Data cards\n");

if(target_material==82) {
fprintf(fp_mcnpx2, "m1 82206 -24.44 82207 -22.94 82208 -52.647\n");
}
if(target_material==8283) {
fprintf(fp_mcnpx2, "m1 82206 -13.625 82207 -12.515 82208 -29.36 83209 -44.5
        \n");
}

if(target_material==74){
fprintf(fp_mcnpx2, "m1 74182 26.62 74183 14.31 74184 30.64 74186 28.43\n");
}
 else if(target_material==83){
fprintf(fp_mcnpx2, "m1 83206 1 \n");
}
if(target_material==92){
fprintf(fp_mcnpx2, "m1 92235 -0.7 92238 -99.3 \n");
}
```

```
    if(target_material==73) {
    fprintf(fp_mcnpx2, "m1 73181 1 \n");
    }

    fprintf(fp_mcnpx2, "m2 92235 -%lf 92238 -%lf 94239 -%lf 93237 -%lf 95241 \n",
            percent_U235_reactor,percent_U238_reactor,percent_Pu_reactor,
            percent_Np_reactor);

    fprintf(fp_mcnpx2,"     -%lf 96243 -%lf 1001 -%lf 8016 -%lf\n",
           percent_Am_reactor,percent_Cm_reactor,percent_H_reactor,
           percent_O2_reactor);


    fprintf(fp_mcnpx2, "m3 26000 -%lf 24000 -%lf\n", percent_weight_Fe,
            percent_weight_Cr);

    fprintf(fp_mcnpx2, "c LINAC beam, Assumed as a circular disc source\n");

    fprintf(fp_mcnpx2, "sdef par=%d pos 0 0 0 axs 0 0 1 rad d2 vec 0 0 1 dir 1
            erg=%lf\n",accelerator_particle, particle_energy);
    fprintf(fp_mcnpx2, "si2 0.48\n");
    fprintf(fp_mcnpx2, "sp2 -21 1\n");
    fprintf(fp_mcnpx2, "c Tallies\n");
    fprintf(fp_mcnpx2, "cut:e j 6.7\n");
    fprintf(fp_mcnpx2, "cut:p j 6.7\n");

    if(accelerator_particle==3) {

     fprintf(fp_mcnpx2, "mode e p n\n");

     /*fprintf(fp_mcnpx2, "phys:p 1000 2j -1\n");*/
     fprintf(fp_mcnpx2, "phys:e 1000 \n");
     fprintf(fp_mcnpx2, "phys:n 1000\n");
    }
    else if(accelerator_particle==9) {

     fprintf(fp_mcnpx2, "mode h n p\n");

     fprintf(fp_mcnpx2, "phys:n 2010 \n");
     fprintf(fp_mcnpx2, "phys:h 2010 \n");
     fprintf(fp_mcnpx2, "phys:p j 1 \n");
     fprintf(fp_mcnpx2, "totnu \n");
   }
/* Tallies (Type f4, Cell Fluence) with Three Energy Bins */
    cell_number = 2;
    skipline   = 1;

     fprintf(fp_mcnpx2, "f4:n");

  for(i=0; i<reactor_Z_cells; ++i) {
    for(j=0; j<reactor_R_cells; ++j) {
     fprintf(fp_mcnpx2, " %d", cell_number);
     ++cell_number;
     if(j>16 && skipline==1){    /* to avoid 80 coulmn problem*/
      fprintf(fp_mcnpx2, "\n    ");
      skipline = 0;
```

```
          }
         }
          fprintf(fp_mcnpx2, "\n    ");
          skipline = 1;
        }
         fprintf(fp_mcnpx2, "e4 0.000001 0.01 %lf\n", particle_energy);
         fprintf(fp_mcnpx2, "print\n");
        /* fprintf(fp_mcnpx2, "nps 10000\n");*/
         fprintf(fp_mcnpx2, "ctme 60\n");

  fclose(fp_mcnpx2);

 }


/***********************************************************
*
*  Subroutine:  make_mcnpx_Neutron2()
*  Purpose:     To Make a MCNPX Input File to Simulate the
*               Neutron Production from the Target
*
***********************************************************/


 int make_mcnpx_Neutron2()  {
/* Target Material was Assumed for the Blanket Also to Avoid Banking Problem, Which Avoids Fission
Neutrons and Hence Banking Problem*/

  int cell_number;
  int skipline;
  double center_r;
  double center_z;

  FILE *fp_mcnpx3;

  fp_mcnpx3 = fopen("C:\\MCNPX2\\BIN\\Neutron2","w");

         fprintf(fp_mcnpx3, "Trial\n");
         fprintf(fp_mcnpx3, "c cell cards\n");

             fprintf(fp_mcnpx3, "1 0   -1      ");
              if(accelerator_particle == 9) {
               fprintf(fp_mcnpx3, "imp:h,n,p = 1     $Vaccume Beam Pipe\n");
              } else if(accelerator_particle == 3) {
               fprintf(fp_mcnpx3, "imp:e,p,n = 1     $Vaccume Beam Pipe\n");
               }
             fprintf(fp_mcnpx3, "c \n");


      for(i=0,cell_number = 2; i<reactor_Z_cells; ++i) {
        for(j=0; j<reactor_R_cells; ++j) {

             if(i<center_cell_z && j==0){
fprintf(fp_mcnpx3, "%d 1 -%lf -%d 1      ", cell_number,density_reactor,
       cell_number,cell_number-1);
   if(accelerator_particle == 9) {
```

```
fprintf(fp_mcnpx3, "imp:h,n,p=1  $Reactor Blanket \n" );
    } else if(accelerator_particle == 3) {
      fprintf(fp_mcnpx3, "imp:e,p,n=1  $Reactor Blanket \n" );
     }
       }
 /* Target */
       else if(i==center_cell_z && j==0){
        if(target_material==73){

          fprintf(fp_mcnpx3, "%d 1 -16.65 -%d     ",cell_number, cell_number);
            if(accelerator_particle == 9) {
              fprintf(fp_mcnpx3,"imp:h,n,p=1  $Target \n");
            } else if(accelerator_particle == 3) {
              fprintf(fp_mcnpx3,"imp:e,p,n=1  $Target \n");
              }
        }
        if(target_material==74) {

          fprintf(fp_mcnpx3, "%d 1 -19.3 -%d      ", cell_number, cell_number);
            if(accelerator_particle == 9) {
              fprintf(fp_mcnpx3,"imp:h,n,p=1  $Target \n");
            } else if(accelerator_particle == 3) {
              fprintf(fp_mcnpx3,"imp:e,p,n=1  $Target \n");
              }

        }
        if(target_material==82) {

          fprintf(fp_mcnpx3, "%d 1 -11.34 -%d     ", cell_number, cell_number);
            if(accelerator_particle == 9) {
              fprintf(fp_mcnpx3,"imp:h,n,p=1  $Target \n");
            } else if(accelerator_particle == 3) {
              fprintf(fp_mcnpx3,"imp:e,p,n=1  $Target \n");
              }

        }
        if(target_material==83) {

          fprintf(fp_mcnpx3, "%d 1 -9.78 -%d      ", cell_number, cell_number);
            if(accelerator_particle == 9) {
              fprintf(fp_mcnpx3,"imp:h,n,p=1  $Target \n");
            } else if(accelerator_particle == 3) {
              fprintf(fp_mcnpx3,"imp:e,p,n=1  $Target \n");
              }

        }
        if(target_material==92) {

          fprintf(fp_mcnpx3, "%d 1 -17.1 -%d      ", cell_number, cell_number);
            if(accelerator_particle == 9) {
              fprintf(fp_mcnpx3,"imp:h,n,p=1  $Target \n");
            } else if(accelerator_particle == 3) {
              fprintf(fp_mcnpx3,"imp:e,p,n=1  $Target \n");
              }

        }
```

```
            }
            else if(j==0){

               fprintf(fp_mcnpx3, "%d 1 -%lf -%d      ", cell_number,density_reactor, cell_number);
                 if(accelerator_particle == 9) {
                   fprintf(fp_mcnpx3,"imp:h,n,p=1  $Reactor Blanket \n");
                 } else if(accelerator_particle == 3) {
                   fprintf(fp_mcnpx3,"imp:e,p,n=1  $Reactor Blanket \n");
                   }

            }

           else {

            fprintf(fp_mcnpx3, "%d 1 -%lf -%d %d      ", cell_number,density_reactor,
cell_number,cell_number-1);
                 if(accelerator_particle == 9) {
                   fprintf(fp_mcnpx3,"imp:h,n,p=1  $Reactor Blanket \n");
                 } else if(accelerator_particle == 3) {
                   fprintf(fp_mcnpx3,"imp:e,p,n=1  $Reactor Blanket \n");
                   }

           }


             ++cell_number;
      }
           fprintf(fp_mcnpx3, "c \n");
           Circum_cell[i] = cell_number-1;

    }

    /* Shielding */
      fprintf(fp_mcnpx3, "%d 3 -7.8 -%d \n", cell_number, cell_number);
        fprintf(fp_mcnpx3,"    ");
        for(i=0; i<reactor_Z_cells; ++i) {
           fprintf(fp_mcnpx3," %d", Circum_cell[i]);
           if(i >= 402) {
            fprintf(fp_mcnpx3,"\n    ");
           }
        }

        if(accelerator_particle == 9) {
          fprintf(fp_mcnpx3,"\n     imp:h,n,p=1  $Steel Shielding\n");
        } else if(accelerator_particle == 3) {
          fprintf(fp_mcnpx3,"\n     imp:e,p,n=1  $Steel Shielding\n");
         }

    /* Vaccume Surroundings */
      fprintf(fp_mcnpx3, "%d 0      %d      ", cell_number+1, cell_number);
        if(accelerator_particle == 9) {
          fprintf(fp_mcnpx3," imp:h,n,p=0  $Vaccume surroundings\n");
        } else if(accelerator_particle == 3) {
          fprintf(fp_mcnpx3," imp:e,p,n=0  $Vaccume surroundings\n");
         }
     fprintf(fp_mcnpx3, "\n");
```

```
    fprintf(fp_mcnpx3,"c Surface cards. \n");
    fprintf(fp_mcnpx3,"c RCC cells \n");


       /* Vaccum Beam Pipe with 0.5 cm Radius*/
        fprintf(fp_mcnpx3, "1 RCC 0 0 0 0 0 %lf 0.5 \n",center_cell_z*dz);
        fprintf(fp_mcnpx3, "c \n");

    cell_number = 2;

  for(i=0; i<reactor_Z_cells; ++i) {
    for(j=0; j<reactor_R_cells; ++j) {
        if(i==center_cell_z && j==0){
  /* Target */
  fprintf(fp_mcnpx3,"%d RCC 0 0 %lf 0 0 %lf %lf
          \n",cell_number,center_cell_z*dz,dz,dr);
        }
        else{
         if(cell[i][j].z_min!=cell[i-1][j].z_max) {
          cell[i][j].z_min = cell[i-1][j].z_max;
         }
         fprintf(fp_mcnpx3,"%d RCC 0 0 %lf 0 0 %lf %lf \n",cell_number,
             cell[i][j].z_min,dz, cell[i][j].r_max);
        }
        ++cell_number;

    }
        fprintf(fp_mcnpx3, "c \n");
  }


 /* Reactor Shielding */
        fprintf(fp_mcnpx3, "%d RCC 0 0 -%lf 0 0 %lf %lf \n", cell_number,
              thickness_shield,reactor_dimension_Z+thickness_shield,
              reactor_dimension_R+thickness_shield);

center_r = reactor_dimension_R/2.00;
center_z = reactor_dimension_Z/2.00;

    fprintf(fp_mcnpx3, "\n");
    fprintf(fp_mcnpx3, "c Data cards\n");

    if(target_material==82) {
    fprintf(fp_mcnpx3, "m1 82206 -24.44 82207 -22.94 82208 -52.647\n");
    }
    if(target_material==8283) {
    fprintf(fp_mcnpx3, "m1 82206 -13.625 82207 -12.515 82208 -29.36 83209 -44.5
          \n");
    }

    if(target_material==74){
    fprintf(fp_mcnpx3, "m1 74182 26.62 74183 14.31 74184 30.64 74186 28.43\n");
    }
     else if(target_material==83){
    fprintf(fp_mcnpx3, "m1 83206 1 \n");
    }
```

```
    if(target_material==92){
    fprintf(fp_mcnpx3, "m1 92235 -0.7 92238 -99.3 \n");
    }
    if(target_material==73) {
    fprintf(fp_mcnpx3, "m1 73181 1 \n");
    }


    fprintf(fp_mcnpx3, "m2 92235 -%lf 92238 -%lf 94239 -%lf 93237 -%lf 95241 \n",
            percent_U235_reactor,percent_U238_reactor,percent_Pu_reactor,
            percent_Np_reactor);

    fprintf(fp_mcnpx3,"    -%lf 96243 -%lf 1001 -%lf 8016 -%lf\n",
            percent_Am_reactor, percent_Cm_reactor,percent_H_reactor,
            percent_O2_reactor);


    fprintf(fp_mcnpx3, "m3 26000 -%lf 24000 -%lf\n", percent_weight_Fe,
            percent_weight_Cr);

    fprintf(fp_mcnpx3, "c LINAC beam, Assumed as a circular disc source\n");

    fprintf(fp_mcnpx3, "sdef par=%d pos 0 0 0 axs 0 0 1 rad d2 vec 0 0 1 dir 1
            erg=%lf\n", accelerator_particle, particle_energy);

    fprintf(fp_mcnpx3, "si2 0.48\n");
    fprintf(fp_mcnpx3, "sp2 -21 1\n");
    fprintf(fp_mcnpx3, "c Tallies\n");
    fprintf(fp_mcnpx3, "cut:e j 6.7\n");
    fprintf(fp_mcnpx3, "cut:p j 6.7\n");

    if(accelerator_particle==3) {

    fprintf(fp_mcnpx3, "mode e p n\n");

    /*fprintf(fp_mcnpx3, "phys:p 1000 2j -1\n");*/

    fprintf(fp_mcnpx3, "phys:e 1000 \n");
    fprintf(fp_mcnpx3, "phys:n 1000\n");
    }
    else if(accelerator_particle==9) {

    fprintf(fp_mcnpx3, "mode h n p\n");

    fprintf(fp_mcnpx3, "phys:n 2010 \n");
    fprintf(fp_mcnpx3, "phys:h 2010 \n");
    fprintf(fp_mcnpx3, "phys:p j 1 \n");
    fprintf(fp_mcnpx3, "totnu \n");
   }

/* Tallies (type f4, cell fluence) with Three Energy Bins */

cell_number = 2;
skipline   = 1;

fprintf(fp_mcnpx3, "f4:n");
```

```
      for(i=0; i<reactor_Z_cells; ++i) {
        for(j=0; j<reactor_R_cells; ++j) {
          fprintf(fp_mcnpx3, " %d", cell_number);
          ++cell_number;
          if(j>16 && skipline==1){    /* to avoid 80 coulmn problem*/
            fprintf(fp_mcnpx3, "\n    ");
            skipline = 0;
          }
        }
          fprintf(fp_mcnpx3, "\n    ");
          skipline = 1;
      }
       fprintf(fp_mcnpx3, "e4 0.000001 0.01 %lf\n",particle_energy);
       fprintf(fp_mcnpx3, "print\n");
       /*fprintf(fp_mcnpx3, "nps 10000\n");*/
       fprintf(fp_mcnpx3, "ctme 120\n");

  fclose(fp_mcnpx3);

 }

void source(double t)  {
  if(((t - time_last_fired) >= accelerator_period) || (t == 0.0))  {
    time_last_fired = t;

    for(L=0; L<N; ++L)  {
      for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k) {
        S[L][k] = Source_flux[L][k];

      }
    }
  }
  else {
    for(L=0; L<N; ++L)  {
      for(k=0; k<NUMBER_OF_ENERGY_GROUPS; ++k) {
         S[L][k] = 0.0;
      }
    }
  }
}


double Read_keff()  {

  char input_line[132];
  char *ptr;
  FILE *fp_keff;

  fp_keff = fopen("okeff", "r");

  fgets(input_line,132, fp_keff);

   while(strstr(input_line, "final result") == NULL) {

      fgets(input_line,132, fp_keff);
```

```
        if(strstr(input_line, "final result") != NULL) {

          ptr = strstr(input_line, "final result");

          ptr = ptr + 17;
          keff = atof(ptr);
          ptr = ptr + 16;
          std_keff = atof(ptr);
        }

    }

    fclose(fp_keff);
}


double Read_Neutron1() {

     FILE *fp_Neutron1;

     fp_Neutron1 = fopen("oSource", "r");

     fgets(input_Neutron, 132, fp_Neutron1);

     while(strstr(input_Neutron, " bank is full.") == NULL) {

       if(strstr(input_Neutron, "tally type 4    track length estimate of particle flux.
         units   1/cm**2") != NULL) {

          break; /* While Loop will Stop Reading Further when it Finds Tallies
                which means No Banking Problem */
       }
       fgets(input_Neutron, 132, fp_Neutron1);

     }
}



double Read_Neutron2()  {



double  Neutrons_per_particle_bin1, Neutrons_per_particle_bin2,
        Neutrons_per_particle_bin3;
char input_Neutron[132];
char *Neuptr;
FILE *fp_Neutron2;

     if(bias == 1) {
       fp_Neutron2 = fopen("oSource2", "r");
     } else {
       fp_Neutron2 = fopen("oSource", "r");
      }
```

```
fgets(input_Neutron, 132, fp_Neutron2);

while(strstr(input_Neutron, "tally type 4    track length estimate of particle flux.     units
      1/cm**2") == NULL)  {

fgets(input_Neutron, 132, fp_Neutron2);

    }

         for(i=0,L=0; i<2000; i++) {
/* There will be 2 Cells Off set From MCNPX to Finite Volumes of the Code */

        fgets(input_Neutron, 132, fp_Neutron2);

       /*Searching for Neutron Production in Each Bin */

          if(strstr(input_Neutron, "energy") != NULL) {

           fgets(input_Neutron, 132, fp_Neutron2);
           Neuptr = strstr(input_Neutron, "   ") ;
           Neuptr = Neuptr + 17;
           Neutronfluence_perparticle[L][0] = atof(Neuptr);

           fgets(input_Neutron, 132, fp_Neutron2);
           Neuptr = strstr(input_Neutron, "   ") ;
           Neuptr = Neuptr + 17;
           Neutronfluence_perparticle[L][1] = atof(Neuptr);

           fgets(input_Neutron, 132, fp_Neutron2);
           Neuptr = strstr(input_Neutron, "   ") ;
           Neuptr = Neuptr + 17;
           Neutronfluence_perparticle[L][2] = atof(Neuptr);

          ++L;
          }

         }
}
```

APPENDIX II

CROSS-SECTION DATA

Nuclear Cross Section Data (in barns)

		Group1(0.569 ev)	Group2(3.17keV)	Group 3(20 MeV)
U235	Cap	8.1	1.61	0.000287
	Escat	14.1	12	3.36
	Fiss	70.4	4.71	1.93
U238	Cap	0.621	4.99	0.001
	Escat	9.27	55.5	3.5
	Fiss	2.57E-06	1.03E-09	1.35
Pu239	Cap	54.9	1.93	0.00181
	Escat	11.6	13.4	3.18
	Fiss	127.41	4.9	2.3
Am241	Cap	3693.42	6.6	0
	Escat	15.15	12.58	2.67
	Fiss	10.76	0.04	2.76
Cm243	Cap	17.38	1.25	0
	Escat	5.4	12.58	0
	Fiss	211.35	8.03	2.5
Cm245	Cap	17.63	0.94	0
	Escat	8.04	10.7	2.99
	Fiss	213.7	7.74	2.15
Np237	Cap	171.14	5.56	0
	Escat	14.7	13.2	3.18
	Fiss	0.01	0.02	2.18
Fe	Cap	0.55	0	0
	Escat	11.4	6.54	0.95
	Fiss	0	0	0
Cr	Cap	0.64	0.03	0
	Escat	4.59	14.6	0.73
	Fiss	0	0	0
H20	Cap	0.05	0	0
	Escat	14.94	14.62	0.66
	Fiss	0	0	0
H	Cap	0.07	0	0
	Escat	20.52	19.96	0.46
	Fiss	0	0	0

Nuclear Cross-Section Data (in barns)				
O	Cap	0	0	0
	Escat	3.9	3.88	1.02
	Fiss	0	0	0
Xe135	T_Abs	14539.71	0.03	0
	Escat	8148	13.22	3.96
	Fiss	0	0	0
I 135	T_Abs	0	0	0
	Escat	4.81	4.81	3.83
	Fiss	0	0	0
Sb135	T_Abs	1.25	0	0
	Escat	4.05	1.8	2.3
	Fiss	0	0	0
Sm 149	T_Abs	489.06	10.47	0.07
	Escat	8.24	21.44	2.63
	Fiss	0	0	0
Pm 149	T_Abs	276.14	14.37	0
	Escat	5.74	15.32	3.97
	Fiss	0	0	0
C 12	T_Abs	0.000721	0.0000126	0.106
	Escat	4.75	4.72	1.03
	Fiss	0	0	0

APPENDIX III

COMPARISON TABLE

Spatial Neutron Flux Distribution in a Steady-State, Finite, Bare Cylindrical Reactor:

Analytical Results:

	0	5	10	15	20	25	30	35	40	45
40	8.82E+09	8.70E+09	8.33E+09	7.74E+09	6.95E+09	5.99E+09	4.90E+09	3.73E+09	2.52E+09	1.32E+09
30	1.64E+10	1.62E+10	1.55E+10	1.44E+10	1.30E+10	1.12E+10	9.14E+09	6.95E+09	4.69E+09	2.45E+09
20	2.25E+10	2.22E+10	2.12E+10	1.97E+10	1.77E+10	1.53E+10	1.25E+10	9.51E+09	6.42E+09	3.36E+09
10	2.64E+10	2.60E+10	2.49E+10	2.31E+10	2.08E+10	1.79E+10	1.47E+10	1.11E+10	7.52E+09	3.93E+09
0	2.77E+10	2.73E+10	2.62E+10	2.43E+10	2.18E+10	1.88E+10	1.54E+10	1.17E+10	7.90E+09	4.13E+09
-10	2.64E+10	2.60E+10	2.49E+10	2.31E+10	2.08E+10	1.79E+10	1.47E+10	1.11E+10	7.52E+09	3.93E+09
-20	2.25E+10	2.22E+10	2.12E+10	1.97E+10	1.77E+10	1.53E+10	1.25E+10	9.51E+09	6.42E+09	3.36E+09
-30	1.64E+10	1.62E+10	1.55E+10	1.44E+10	1.30E+10	1.12E+10	9.14E+09	6.95E+09	4.69E+09	2.45E+09
-40	8.82E+09	8.70E+09	8.33E+09	7.74E+09	6.95E+09	5.99E+09	4.90E+09	3.73E+09	2.52E+09	1.32E+09

Numerical Results:

	0	5	10	15	20	25	30	35	40	45
40	1.25E+10	1.23E+10	1.18E+10	1.11E+10	1.01E+10	8.96E+09	7.66E+09	6.19E+09	4.53E+09	2.55E+09
30	1.78E+10	1.75E+10	1.67E+10	1.55E+10	1.41E+10	1.24E+10	1.06E+10	8.50E+09	6.18E+09	3.46E+09
20	2.28E+10	2.22E+10	2.10E+10	1.94E+10	1.75E+10	1.54E+10	1.30E+10	1.04E+10	7.54E+09	4.22E+09
10	2.64E+10	2.56E+10	2.41E+10	2.22E+10	1.99E+10	1.73E+10	1.46E+10	1.16E+10	8.41E+09	4.69E+09
0	2.77E+10	2.68E+10	2.51E+10	2.30E+10	2.06E+10	1.79E+10	1.50E+10	1.20E+10	8.64E+09	4.82E+09
-10	2.59E+10	2.51E+10	2.37E+10	2.17E+10	1.94E+10	1.69E+10	1.42E+10	1.13E+10	8.18E+09	4.57E+09
-20	2.18E+10	2.11E+10	2.00E+10	1.84E+10	1.65E+10	1.45E+10	1.22E+10	9.72E+09	7.04E+09	3.93E+09
-30	1.55E+10	1.49E+10	1.42E+10	1.32E+10	1.19E+10	1.05E+10	8.90E+09	7.16E+09	5.22E+09	2.94E+09
-40	6.47E+09	5.17E+09	5.61E+09	5.60E+09	5.31E+09	4.83E+09	4.22E+09	3.49E+09	2.62E+09	1.60E+09

Comparison between Analytical and Numerical Results for an Infinite Cylindrical Reactor with Reflector:

Radial Distance (cm)	Analytical	Numerical	%Error
0	2.57E+10	2.574E+10	0.01
5	2.54E+10	2.498E+10	1.5
10	2.43E+10	2.355E+10	3.0
15	2.25E+10	2.161E+10	3.9
20	2.01E+10	1.927E+10	4.2
25	1.72E+10	1.659E+10	3.8
30	1.40E+10	1.363E+10	2.6
35	1.05E+10	1.035E+10	1.3
40	6.90E+09	6.659E+09	3.5
45	1.31E+09	1.286E+09	1.9

Transient Neutron Flux in a Finite Bare Cylindrical Reactor:

Time (sec)	Analytical	Numerical	%Error
0	1.00E+13	1.00E+13	0.0
0.0001	9.87E+12	9.25E+12	6.3
0.0002	8.69E+12	7.9E+12	9.1
0.0004	5.22E+12	4.93E+12	5.6
0.0006	2.88E+12	2.8E+12	2.7
0.0008	1.56E+12	1.54E+12	1.5
0.001	8.47E+11	8.37E+11	1.2
0.0015	1.83E+11	1.81E+11	1.2
0.002	3.94E+10	3.89E+10	1.4
0.0025	8.52E+09	8.37E+09	1.7
0.003	1.84E+09	1.8E+09	1.9
0.0035	3.97E+08	3.88E+08	2.1
0.004	8.56E+07	83629666	2.3
0.0045	1.85E+07	18009935	2.6
0.005	3.99E+06	3878511	2.8
0.0055	8.61E+05	835254.4	3.0
0.006	1.86E+05	179876.3	-5.6

Neutron Poison Concentration after Reactor Shutdown:

Comparison between ^{135}Xe Number Densities from Numerical and Analytical Results, after Reactor Shutdown at t = 0

Time (sec)	Analytical (n/cm^3)	Numerical (n/cm^3)	% Error
0	2.09E+19	2.09E+19	0
200	2.25E+19	2.28E+19	-1.2
400	2.40E+19	2.45E+19	-2.0
600	2.55E+19	2.62E+19	-2.6
800	2.70E+19	2.79E+19	-3.0
1000	2.85E+19	2.95E+19	-3.3
1200	3.00E+19	3.10E+19	-3.4
1400	3.14E+19	3.25E+19	-3.6

1600	3.28E+19	3.40E+19	-3.4
1800	3.42E+19	3.53E+19	-3.3

Numerical Comparisons:

Cell Density Comparisons:

Time Dependent Neutron Flux for Different Cell Densities

10by10		% Error
0	2.36E+10	10.5
0.0001	1.78E+09	6.3
0.0002	55048150	3.2
0.0003	9714555	1.0
0.0004	8727815	1.1
0.0005	8707979	1.1
0.0006	8708893	1.1
0.0007	8710250	1.1
0.0008	8711616	1.1
0.0009	8712983	1.1
0.001	8714351	1.1
0.0011	8715719	1.1
0.0012	8717087	1.1
0.0013	8718456	1.1
0.0014	8719825	1.1
0.0015	8721195	1.1
0.0016	8722565	1.1
0.0017	8723936	1.1
0.0018	8725306	1.1
0.0019	8726678	1.1
0.002	8728049	1.1
0.0021	8729421	1.1
0.0022	8730794	1.1
0.0023	8732167	1.1
0.0024	8733540	1.1
0.0025	8734914	1.1
0.0026	8736288	1.1
0.0027	8737662	1.0
0.0028	8739037	1.0
0.0029	8740412	1.0
0.003	8741788	1.0
0.0031	8743164	1.0
0.0032	8744541	1.0
0.0033	8745918	1.0
0.0034	8747295	1.0

0.0035	8748673	1.0
0.0036	8750051	1.0
0.0037	8751430	1.0
0.0038	8752809	1.0
0.0039	8754188	1.0
0.004	8755568	1.0
0.0041	8756948	1.0
0.0042	8758329	1.0
0.0043	8759710	1.0
0.0044	8761091	1.0
0.0045	8762473	1.0
0.0046	8763855	1.0
0.0047	8765238	1.0
0.0048	8766621	1.0
0.0049	8768004	1.0
0.005	8769388	1.0
0.0051	8770772	1.0
0.0052	8772157	1.0
0.0053	8773542	1.0
0.0054	8774928	1.0
0.0055	8776314	1.0
0.0056	8777700	1.0
0.0057	8779087	1.0
0.0058	8780474	1.0
0.0059	8781861	1.0
0.006	8783249	1.0
0.0061	8784638	0.9
0.0062	8786027	0.9
0.0063	8787416	0.9
0.0064	8788805	0.9
0.0065	8790195	0.9
0.0066	8791586	0.9
0.0067	8792977	0.9
0.0068	8794368	0.9
0.0069	8795759	0.9
0.007	8797151	0.9
0.0071	8798544	0.9
0.0072	8799937	0.9
0.0073	8801330	0.9
0.0074	8802724	0.9
0.0075	8804118	0.9
0.0076	8805512	0.9
0.0077	8806907	0.9
0.0078	8808302	0.9
0.0079	8809698	0.9
0.008	8811094	0.9
0.0081	8812491	0.9

0.0082	8813888	0.9
0.0083	8815285	0.9
0.0084	8816683	0.9
0.0085	8818081	0.9
0.0086	8819480	0.9
0.0087	8820879	0.9
0.0088	8822278	0.9
0.0089	8823678	0.9
0.009	8825078	0.9
0.0091	8826479	0.9
0.0092	8827880	0.9
0.0093	8829281	0.9
0.0094	8830683	0.9
0.0095	8832086	0.8
0.0096	8833488	0.8
0.0097	8834891	0.8
0.0098	8836295	0.8
0.0099	8837699	0.8
0.01	8839103	0.8
0.0101	8840508	0.8
0.0102	8841913	0.8
0.0103	8843319	0.8
0.0104	8844725	0.8
0.0105	8846131	0.8
0.0106	8847538	0.8
0.0107	8848945	0.8
0.0108	8850353	0.8
0.0109	8851761	0.8
0.011	8853169	0.8
0.0111	8854578	0.8
0.0112	8855987	0.8
0.0113	8857397	0.8
0.0114	8858807	0.8
0.0115	8860217	0.8
0.0116	8861628	0.8
0.0117	8863040	0.8
0.0118	8864451	0.8
0.0119	8865863	0.8
0.012	8867276	0.8
0.0121	8868689	0.8
0.0122	8870102	0.8
0.0123	8871516	0.8
0.0124	8872930	0.8
0.0125	8874345	0.8
0.0126	8875760	0.8
0.0127	8877175	0.8
0.0128	8878591	0.8

0.0129	8880007	0.7
0.013	8881424	0.7
0.0131	8882841	0.7
0.0132	8884259	0.7
0.0133	8885677	0.7
0.0134	8887095	0.7
0.0135	8888514	0.7
0.0136	8889933	0.7
0.0137	8891352	0.7
0.0138	8892772	0.7
0.0139	8894193	0.7
0.014	8895613	0.7
0.0141	8897035	0.7
0.0142	8898456	0.7
0.0143	8899878	0.7
0.0144	8901301	0.7
0.0145	8902724	0.7
0.0146	8904147	0.7
0.0147	8905570	0.7
0.0148	8906995	0.7
0.0149	8908419	0.7
0.015	8909844	0.7
0.0151	8911269	0.7
0.0152	8912695	0.7
0.0153	8914121	0.7
0.0154	8915548	0.7
0.0155	8916975	0.7
0.0156	8918402	0.7
0.0157	8919830	0.7
0.0158	8921258	0.7
0.0159	8922687	0.7
0.016	8924116	0.7

Timestep Comparisons:

10 by 10 Time step=0.05 milli-sec										
	0	5	10	15	20	25	30	35	40	45
50	7.34E+05	7.13E+05	6.72E+05	6.13E+05	5.37E+05	4.48E+05	3.49E+05	2.47E+05	1.52E+05	6.93E+04
40	3.91E+06	3.80E+06	3.58E+06	3.26E+06	2.85E+06	2.35E+06	1.80E+06	1.19E+06	4.28E+05	1.81E+05
30	6.29E+06	6.11E+06	5.75E+06	5.23E+06	4.56E+06	3.76E+06	2.86E+06	1.88E+06	6.70E+05	2.82E+05
20	8.08E+06	7.84E+06	7.38E+06	6.70E+06	5.84E+06	4.81E+06	3.65E+06	2.40E+06	8.52E+05	3.58E+05
10	9.18E+06	8.90E+06	8.37E+06	7.60E+06	6.61E+06	5.44E+06	4.13E+06	2.71E+06	9.62E+05	4.03E+05
0	9.45E+06	9.17E+06	8.62E+06	7.82E+06	6.80E+06	5.60E+06	4.24E+06	2.78E+06	9.87E+05	4.14E+05
-10	8.82E+06	8.55E+06	8.04E+06	7.29E+06	6.34E+06	5.22E+06	3.96E+06	2.60E+06	9.21E+05	3.87E+05
-20	7.27E+06	7.05E+06	6.63E+06	6.01E+06	5.23E+06	4.31E+06	3.27E+06	2.15E+06	7.63E+05	3.20E+05
-30	4.84E+06	4.70E+06	4.41E+06	3.99E+06	3.47E+06	2.86E+06	2.17E+06	1.43E+06	5.17E+05	2.19E+05
-40	1.11E+06	1.08E+06	1.01E+06	8.97E+05	7.67E+05	6.26E+05	4.81E+05	3.37E+05	2.06E+05	9.36E+04

10 by 10 Time step = 0.1 milli-sec										
	0	5	10	15	20	25	30	35	40	45
50	7.54E+05	7.32E+05	6.91E+05	6.30E+05	5.52E+05	4.61E+05	3.60E+05	2.56E+05	1.66E+05	7.58E+04
40	3.89E+06	3.78E+06	3.56E+06	3.24E+06	2.83E+06	2.35E+06	1.79E+06	1.18E+06	5.09E+05	2.13E+05
30	6.25E+06	6.07E+06	5.72E+06	5.20E+06	4.54E+06	3.74E+06	2.85E+06	1.87E+06	7.95E+05	3.30E+05
20	8.02E+06	7.78E+06	7.32E+06	6.65E+06	5.80E+06	4.78E+06	3.63E+06	2.38E+06	1.01E+06	4.18E+05
10	9.09E+06	8.82E+06	8.29E+06	7.53E+06	6.55E+06	5.40E+06	4.09E+06	2.69E+06	1.13E+06	4.70E+05
0	9.34E+06	9.07E+06	8.52E+06	7.73E+06	6.72E+06	5.54E+06	4.20E+06	2.75E+06	1.16E+06	4.82E+05
-10	8.70E+06	8.44E+06	7.94E+06	7.20E+06	6.26E+06	5.16E+06	3.91E+06	2.57E+06	1.08E+06	4.50E+05
-20	7.17E+06	6.96E+06	6.54E+06	5.93E+06	5.16E+06	4.25E+06	3.23E+06	2.12E+06	9.00E+05	3.73E+05
-30	4.77E+06	4.63E+06	4.35E+06	3.94E+06	3.42E+06	2.82E+06	2.14E+06	1.42E+06	6.08E+05	2.54E+05
-40	1.12E+06	1.09E+06	1.02E+06	9.06E+05	7.75E+05	6.34E+05	4.88E+05	3.44E+05	2.20E+05	9.99E+04

Time (sec)	0.1 Time Step	0.05 Time Step	% Error
0	0	0	0
0.0001	1.15E+10	1.33E+09	88.4
0.0002	5.33E+08	3.00E+06	99.4
0.0003	11739777	1.63E+06	86.1
0.0004	1788993	1.63E+06	9.1
0.0005	1628892	1.63E+06	0.2
0.0006	1626233	1.63E+06	0.0
0.0007	1626199	1.63E+06	0.0
0.0008	1626213	1.63E+06	0.0
0.0009	1626228	1.63E+06	0.0
0.001	1626243	1.63E+06	0.0
0.0011	1626258	1.63E+06	0.0
0.0012	1626273	1.63E+06	0.0
0.0013	1626288	1.63E+06	0.0
0.0014	1626303	1.63E+06	0.0
0.0015	1626318	1.63E+06	0.0
0.0016	1626333	1.63E+06	0.0
0.0017	1626348	1.63E+06	0.0
0.0018	1626364	1.63E+06	0.0
0.0019	1626379	1.63E+06	0.0
0.002	1626394	1.63E+06	0.0
0.0021	1626410	1.63E+06	0.0
0.0022	1626425	1.63E+06	0.0
0.0023	1626441	1.63E+06	0.0
0.0024	1626456	1.63E+06	0.0
0.0025	1626472	1.63E+06	0.0
0.0026	1626487	1.63E+06	0.0
0.0027	1626503	1.63E+06	0.0

Neutron Flux Distribution from 20 by 20 Cell Density										
Axial Distance (cm)	Radial Distance (cm)									
	0	2.5	5	7.5	10	12.5	15	17.5	20	22.5
50	5.9E+08	5.87E+08	5.8E+08	5.7E+08	5.57E+08	5.41E+08	5.22E+08	5E+08	4.75E+08	4.47E+08
45	1.08E+09	1.08E+09	1.06E+09	1.04E+09	1.02E+09	9.89E+08	9.52E+08	9.09E+08	8.61E+08	8.06E+08
40	5.53E+08	5.49E+08	5.42E+08	5.31E+08	5.16E+08	4.97E+08	4.75E+08	4.49E+08	4.19E+08	3.86E+08
35	8.45E+08	8.39E+08	8.28E+08	8.1E+08	7.87E+08	7.58E+08	7.23E+08	6.83E+08	6.36E+08	5.85E+08
30	1.09E+09	1.09E+09	1.07E+09	1.05E+09	1.02E+09	9.78E+08	9.32E+08	8.79E+08	8.19E+08	7.51E+08
25	1.31E+09	1.3E+09	1.28E+09	1.25E+09	1.21E+09	1.17E+09	1.11E+09	1.05E+09	9.74E+08	8.93E+08
20	1.5E+09	1.48E+09	1.46E+09	1.43E+09	1.39E+09	1.33E+09	1.27E+09	1.19E+09	1.11E+09	1.01E+09
15	1.66E+09	1.64E+09	1.62E+09	1.58E+09	1.53E+09	1.47E+09	1.4E+09	1.32E+09	1.22E+09	1.12E+09
10	1.79E+09	1.78E+09	1.75E+09	1.71E+09	1.65E+09	1.58E+09	1.51E+09	1.41E+09	1.31E+09	1.2E+09
5	1.89E+09	1.87E+09	1.84E+09	1.8E+09	1.74E+09	1.67E+09	1.58E+09	1.48E+09	1.38E+09	1.26E+09
0	1.95E+09	1.93E+09	1.9E+09	1.85E+09	1.79E+09	1.71E+09	1.62E+09	1.52E+09	1.41E+09	1.29E+09
-5	1.95E+09	1.93E+09	1.9E+09	1.85E+09	1.79E+09	1.71E+09	1.62E+09	1.52E+09	1.41E+09	1.29E+09
-10	1.9E+09	1.88E+09	1.85E+09	1.8E+09	1.74E+09	1.67E+09	1.58E+09	1.48E+09	1.37E+09	1.25E+09
-15	1.79E+09	1.78E+09	1.75E+09	1.7E+09	1.65E+09	1.58E+09	1.5E+09	1.4E+09	1.3E+09	1.19E+09
-20	1.63E+09	1.62E+09	1.59E+09	1.55E+09	1.5E+09	1.44E+09	1.36E+09	1.28E+09	1.18E+09	1.08E+09
-25	1.4E+09	1.39E+09	1.37E+09	1.34E+09	1.29E+09	1.24E+09	1.18E+09	1.11E+09	1.03E+09	9.37E+08
-30	1.11E+09	1.11E+09	1.09E+09	1.06E+09	1.03E+09	9.85E+08	9.36E+08	8.8E+08	8.17E+08	7.48E+08
-35	7.44E+08	7.38E+08	7.26E+08	7.09E+08	6.87E+08	6.59E+08	6.27E+08	5.9E+08	5.49E+08	5.03E+08
-40	1.61E+09	1.6E+09	1.57E+09	1.53E+09	1.49E+09	1.43E+09	1.37E+09	1.29E+09	1.21E+09	1.13E+09
-45	8.95E+08	8.88E+08	8.73E+08	8.53E+08	8.27E+08	7.96E+08	7.61E+08	7.23E+08	6.8E+08	6.35E+08

Neutron Flux Distribution from 20 by 20 Cell Density										
Axial Distance (cm)	Radial Distance (cm)									
	25	27.5	30	32.5	35	37.5	40	42.5	45	47.5
50	4.16E+08	3.83E+08	3.48E+08	3.11E+08	2.75E+08	2.39E+08	2.02E+08	1.61E+08	1.15E+08	61747573
45	7.45E+08	6.79E+08	6.08E+08	5.35E+08	4.68E+08	4.21E+08	3.59E+08	2.86E+08	2.02E+08	1.07E+08
40	3.49E+08	3.08E+08	2.64E+08	2.15E+08	1.62E+08	4.48E+08	4.72E+08	3.94E+08	2.81E+08	1.48E+08
35	5.27E+08	4.64E+08	3.94E+08	3.18E+08	2.37E+08	5.9E+08	6.14E+08	5.08E+08	3.58E+08	1.87E+08
30	6.76E+08	5.93E+08	5.03E+08	4.04E+08	3E+08	7.22E+08	7.45E+08	6.12E+08	4.29E+08	2.23E+08
25	8.02E+08	7.03E+08	5.95E+08	4.77E+08	3.54E+08	8.37E+08	8.62E+08	7.06E+08	4.93E+08	2.56E+08
20	9.11E+08	7.97E+08	6.74E+08	5.4E+08	4E+08	9.38E+08	9.64E+08	7.88E+08	5.5E+08	2.85E+08
15	1E+09	8.76E+08	7.4E+08	5.93E+08	4.38E+08	1.03E+09	1.05E+09	8.58E+08	5.98E+08	3.1E+08
10	1.07E+09	9.39E+08	7.92E+08	6.34E+08	4.69E+08	1.09E+09	1.12E+09	9.15E+08	6.37E+08	3.29E+08
5	1.13E+09	9.83E+08	8.29E+08	6.63E+08	4.9E+08	1.15E+09	1.17E+09	9.56E+08	6.66E+08	3.44E+08
0	1.15E+09	1.01E+09	8.48E+08	6.78E+08	5.01E+08	1.17E+09	1.2E+09	9.78E+08	6.81E+08	3.52E+08
-5	1.15E+09	1.01E+09	8.48E+08	6.78E+08	5E+08	1.18E+09	1.2E+09	9.8E+08	6.82E+08	3.52E+08
-10	1.12E+09	9.8E+08	8.26E+08	6.61E+08	4.88E+08	1.15E+09	1.18E+09	9.59E+08	6.67E+08	3.45E+08
-15	1.06E+09	9.27E+08	7.82E+08	6.26E+08	4.62E+08	1.1E+09	1.12E+09	9.16E+08	6.37E+08	3.29E+08
-20	9.69E+08	8.47E+08	7.14E+08	5.72E+08	4.23E+08	1.01E+09	1.04E+09	8.5E+08	5.92E+08	3.06E+08
-25	8.41E+08	7.35E+08	6.22E+08	4.99E+08	3.7E+08	9.03E+08	9.3E+08	7.61E+08	5.32E+08	2.76E+08
-30	6.72E+08	5.89E+08	4.99E+08	4.02E+08	2.99E+08	7.58E+08	7.87E+08	6.48E+08	4.55E+08	2.37E+08
-35	4.53E+08	3.99E+08	3.4E+08	2.76E+08	2.09E+08	5.88E+08	6.17E+08	5.14E+08	3.64E+08	1.91E+08
-40	1.03E+09	9.34E+08	8.3E+08	7.26E+08	6.3E+08	5.63E+08	4.78E+08	3.79E+08	2.67E+08	1.41E+08
-45	5.86E+08	5.35E+08	4.81E+08	4.28E+08	3.75E+08	3.24E+08	2.72E+08	2.16E+08	1.54E+08	82179050

Neutron Flux Distribution from 10 by 10 Cell Density										
	Radial Distance (cm)									
Axial Distance (cm)	0	5	10	15	20	25	30	35	40	45
50	9.04E+08	8.85E+08	8.47E+08	7.9E+08	7.16E+08	6.25E+08	5.2E+08	4.05E+08	2.95E+08	1.64E+08
40	7.81E+08	7.61E+08	7.21E+08	6.62E+08	5.85E+08	4.9E+08	3.77E+08	2.47E+08	5.59E+08	3.24E+08
30	1.25E+09	1.21E+09	1.15E+09	1.05E+09	9.23E+08	7.7E+08	5.89E+08	3.81E+08	8.1E+08	4.6E+08
20	1.6E+09	1.55E+09	1.46E+09	1.34E+09	1.17E+09	9.73E+08	7.42E+08	4.78E+08	1E+09	5.68E+08
10	1.83E+09	1.77E+09	1.67E+09	1.52E+09	1.33E+09	1.1E+09	8.36E+08	5.38E+08	1.12E+09	6.34E+08
0	1.89E+09	1.84E+09	1.72E+09	1.57E+09	1.37E+09	1.13E+09	8.61E+08	5.54E+08	1.16E+09	6.53E+08
-10	1.78E+09	1.72E+09	1.62E+09	1.47E+09	1.29E+09	1.07E+09	8.12E+08	5.23E+08	1.09E+09	6.17E+08
-20	1.48E+09	1.44E+09	1.36E+09	1.24E+09	1.08E+09	8.97E+08	6.84E+08	4.41E+08	9.31E+08	5.27E+08
-30	9.95E+08	9.66E+08	9.1E+08	8.29E+08	7.27E+08	6.04E+08	4.63E+08	3.01E+08	6.65E+08	3.84E+08
-40	1.07E+09	9.2E+08	9.27E+08	8.87E+08	8.11E+08	7.1E+08	5.9E+08	4.6E+08	3.38E+08	1.94E+08

APPENDIX IV

SAMPLE SIMULATION SUMMARY OUTPUT FILE DEVELOPED BY CODE

```
THE SIMULATION RESULTS ALONG WITH THE IMPORTANT INPUT SPECIFICATIONS
ARE SUMMARISED IN THE FILE

The accelerator coupled reactor was analyzed for the transient
behavior.
Time step of the simulations      : 0.000100 sec

***********************************
*   Program Input Details          *
***********************************


A.  Reactor Description

As a Test Case: The subcritical blanket is filed with uranium,
plutonium oxides and some minor actinides.
The volumetric percent of core and water in the reactor are 0.900000
and 0.100000 respectively.
The percent by weight of Cm, Am, UO2, PuO2, NpO2 are 0.005000 0.200000
0.770000 0.005000 0.020000 respectively.
The reactor is shielded by steel reflector (Fe (0.900000),
Cr(0.100000)).

In this analysis neutron transport/diffusion equations were used which
are solved by finite volume method.
    Temperature (K)                        : 300.000000
    Radius of the reactor                  : 50 cm
    Finite volumes in the radial direction: 10
    Axial dimension of the reactor         : 100 cm
    Finite volumes in the axial direction : 10

Three neutron energy groups (thermal = 0.569826 ev,
epithermal=3177.500000 ev, fast=20300000.000000ev) were assumed.

All the fission and delayed neutrons were assumed to be produced in the
fast group.

All the neutron interaction cross sections for this analysis were
obtained from ENDFB libraries of MCNPX.

B.  Accelerator Description
    1.  Accelerator Particle       :9 (3=electron;9=proton)
    2.  Particle Energy (MeV)      :1000.000000
```

```
3.  Accelerator Current (A)     :0.001000
    4.  Acce. Frequency (Hz)        :60.000000
    5.  Acc. Duty Cycles (micros)   :2.000000


************************************
*   Program Output                 *
************************************


The blanket was analyzed for criticality using MCNPX
 Keff = 0.682490  std_keff = 0.001160, Reactor is sub critical

MCNPX simulations were done to find the neutron production in the
target and the subcritical blanket.

Speed of the neutrons in each group are:
     speed[0] : 1043982.721176 cm/sec
     speed[1] : 77958766.649428 cm/sec
     speed[2] : 6231178596.381267 cm/sec
The diffusion coefficient and diffusion length in the reactor and
reflector are:
   Diffusion coefficient in the core :
              Thermal     :  0.795446 cm
              Epithermal  :  0.282186 cm
              Fast        :  2.890165 cm
   Diffusion coefficient in the reflector :
              Thermal     :  0.371582 cm
              Epithermal  :  0.543755 cm
              Fast        :  4.294841 cm
   Diffusion length in the core :
              Thermal     :  0.207788 cm
              Epithermal  :  1.521566 cm
              Fast        :  8.709209 cm
   Diffusion length in the reflector :
              Thermal     :  2.803170 cm
              Epithermal  :  46.650384 cm
              Fast        :  inf cm
Time elapsed for the 3600.000000 sec of reactor simulation = 141.807778
hours
```

APPENDIX V

SAMPLE MCNPX INPUT FLIE FOR NEUTRON PRODUCTION

```
Neutron
c cell cards
1 0  -1     imp:h,n,p = 1    $Vaccume Beam Pipe
c
2 2 -1.997000 -2 1     imp:h,n,p=1  $Reactor Blanket
3 2 -1.997000 -3 2      imp:h,n,p=1  $Reactor Blanket
4 2 -1.997000 -4 3      imp:h,n,p=1  $Reactor Blanket
5 2 -1.997000 -5 4      imp:h,n,p=1  $Reactor Blanket
6 2 -1.997000 -6 5      imp:h,n,p=1  $Reactor Blanket
7 2 -1.997000 -7 6      imp:h,n,p=1  $Reactor Blanket
8 2 -1.997000 -8 7      imp:h,n,p=1  $Reactor Blanket
9 2 -1.997000 -9 8      imp:h,n,p=1  $Reactor Blanket
10 2 -1.997000 -10 9      imp:h,n,p=1  $Reactor Blanket
11 2 -1.997000 -11 10      imp:h,n,p=1  $Reactor Blanket
c
12 2 -1.997000 -12 1     imp:h,n,p=1  $Reactor Blanket
13 2 -1.997000 -13 12      imp:h,n,p=1  $Reactor Blanket
14 2 -1.997000 -14 13      imp:h,n,p=1  $Reactor Blanket
15 2 -1.997000 -15 14      imp:h,n,p=1  $Reactor Blanket
16 2 -1.997000 -16 15      imp:h,n,p=1  $Reactor Blanket
17 2 -1.997000 -17 16      imp:h,n,p=1  $Reactor Blanket
18 2 -1.997000 -18 17      imp:h,n,p=1  $Reactor Blanket
19 2 -1.997000 -19 18      imp:h,n,p=1  $Reactor Blanket
20 2 -1.997000 -20 19      imp:h,n,p=1  $Reactor Blanket
21 2 -1.997000 -21 20      imp:h,n,p=1  $Reactor Blanket
c
22 2 -1.997000 -22 1     imp:h,n,p=1  $Reactor Blanket
23 2 -1.997000 -23 22      imp:h,n,p=1  $Reactor Blanket
24 2 -1.997000 -24 23      imp:h,n,p=1  $Reactor Blanket
25 2 -1.997000 -25 24      imp:h,n,p=1  $Reactor Blanket
26 2 -1.997000 -26 25      imp:h,n,p=1  $Reactor Blanket
27 2 -1.997000 -27 26      imp:h,n,p=1  $Reactor Blanket
28 2 -1.997000 -28 27      imp:h,n,p=1  $Reactor Blanket
29 2 -1.997000 -29 28      imp:h,n,p=1  $Reactor Blanket
30 2 -1.997000 -30 29      imp:h,n,p=1  $Reactor Blanket
31 2 -1.997000 -31 30      imp:h,n,p=1  $Reactor Blanket
c
32 2 -1.997000 -32 1     imp:h,n,p=1  $Reactor Blanket
33 2 -1.997000 -33 32      imp:h,n,p=1  $Reactor Blanket
34 2 -1.997000 -34 33      imp:h,n,p=1  $Reactor Blanket
```

```
35 2 -1.997000 -35 34      imp:h,n,p=1  $Reactor Blanket
36 2 -1.997000 -36 35      imp:h,n,p=1  $Reactor Blanket
37 2 -1.997000 -37 36      imp:h,n,p=1  $Reactor Blanket
38 2 -1.997000 -38 37      imp:h,n,p=1  $Reactor Blanket
39 2 -1.997000 -39 38      imp:h,n,p=1  $Reactor Blanket
40 2 -1.997000 -40 39      imp:h,n,p=1  $Reactor Blanket
41 2 -1.997000 -41 40      imp:h,n,p=1  $Reactor Blanket
c
42 1 -11.34 -42    imp:h,n,p=1  $Target
43 2 -1.997000 -43 42      imp:h,n,p=1  $Reactor Blanket
44 2 -1.997000 -44 43      imp:h,n,p=1  $Reactor Blanket
45 2 -1.997000 -45 44      imp:h,n,p=1  $Reactor Blanket
46 2 -1.997000 -46 45      imp:h,n,p=1  $Reactor Blanket
47 2 -1.997000 -47 46      imp:h,n,p=1  $Reactor Blanket
48 2 -1.997000 -48 47      imp:h,n,p=1  $Reactor Blanket
49 2 -1.997000 -49 48      imp:h,n,p=1  $Reactor Blanket
50 2 -1.997000 -50 49      imp:h,n,p=1  $Reactor Blanket
51 2 -1.997000 -51 50      imp:h,n,p=1  $Reactor Blanket
c
52 2 -1.997000 -52     imp:h,n,p=1  $Reactor Blanket
53 2 -1.997000 -53 52      imp:h,n,p=1  $Reactor Blanket
54 2 -1.997000 -54 53      imp:h,n,p=1  $Reactor Blanket
55 2 -1.997000 -55 54      imp:h,n,p=1  $Reactor Blanket
56 2 -1.997000 -56 55      imp:h,n,p=1  $Reactor Blanket
57 2 -1.997000 -57 56      imp:h,n,p=1  $Reactor Blanket
58 2 -1.997000 -58 57      imp:h,n,p=1  $Reactor Blanket
59 2 -1.997000 -59 58      imp:h,n,p=1  $Reactor Blanket
60 2 -1.997000 -60 59      imp:h,n,p=1  $Reactor Blanket
61 2 -1.997000 -61 60      imp:h,n,p=1  $Reactor Blanket
c
62 2 -1.997000 -62     imp:h,n,p=1  $Reactor Blanket
63 2 -1.997000 -63 62      imp:h,n,p=1  $Reactor Blanket
64 2 -1.997000 -64 63      imp:h,n,p=1  $Reactor Blanket
65 2 -1.997000 -65 64      imp:h,n,p=1  $Reactor Blanket
66 2 -1.997000 -66 65      imp:h,n,p=1  $Reactor Blanket
67 2 -1.997000 -67 66      imp:h,n,p=1  $Reactor Blanket
68 2 -1.997000 -68 67      imp:h,n,p=1  $Reactor Blanket
69 2 -1.997000 -69 68      imp:h,n,p=1  $Reactor Blanket
70 2 -1.997000 -70 69      imp:h,n,p=1  $Reactor Blanket
71 2 -1.997000 -71 70      imp:h,n,p=1  $Reactor Blanket
c
72 2 -1.997000 -72     imp:h,n,p=1  $Reactor Blanket
73 2 -1.997000 -73 72      imp:h,n,p=1  $Reactor Blanket
74 2 -1.997000 -74 73      imp:h,n,p=1  $Reactor Blanket
75 2 -1.997000 -75 74      imp:h,n,p=1  $Reactor Blanket
76 2 -1.997000 -76 75      imp:h,n,p=1  $Reactor Blanket
77 2 -1.997000 -77 76      imp:h,n,p=1  $Reactor Blanket
78 2 -1.997000 -78 77      imp:h,n,p=1  $Reactor Blanket
79 2 -1.997000 -79 78      imp:h,n,p=1  $Reactor Blanket
80 2 -1.997000 -80 79      imp:h,n,p=1  $Reactor Blanket
81 2 -1.997000 -81 80      imp:h,n,p=1  $Reactor Blanket
c
82 2 -1.997000 -82     imp:h,n,p=1  $Reactor Blanket
83 2 -1.997000 -83 82      imp:h,n,p=1  $Reactor Blanket
84 2 -1.997000 -84 83      imp:h,n,p=1  $Reactor Blanket
85 2 -1.997000 -85 84      imp:h,n,p=1  $Reactor Blanket
```

```
86 2 -1.997000 -86 85      imp:h,n,p=1  $Reactor Blanket
87 2 -1.997000 -87 86      imp:h,n,p=1  $Reactor Blanket
88 2 -1.997000 -88 87      imp:h,n,p=1  $Reactor Blanket
89 2 -1.997000 -89 88      imp:h,n,p=1  $Reactor Blanket
90 2 -1.997000 -90 89      imp:h,n,p=1  $Reactor Blanket
91 2 -1.997000 -91 90      imp:h,n,p=1  $Reactor Blanket
c
92 2 -1.997000 -92     imp:h,n,p=1  $Reactor Blanket
93 2 -1.997000 -93 92      imp:h,n,p=1  $Reactor Blanket
94 2 -1.997000 -94 93      imp:h,n,p=1  $Reactor Blanket
95 2 -1.997000 -95 94      imp:h,n,p=1  $Reactor Blanket
96 2 -1.997000 -96 95      imp:h,n,p=1  $Reactor Blanket
97 2 -1.997000 -97 96      imp:h,n,p=1  $Reactor Blanket
98 2 -1.997000 -98 97      imp:h,n,p=1  $Reactor Blanket
99 2 -1.997000 -99 98      imp:h,n,p=1  $Reactor Blanket
100 2 -1.997000 -100 99      imp:h,n,p=1  $Reactor Blanket
101 2 -1.997000 -101 100      imp:h,n,p=1  $Reactor Blanket
c
102 3 -7.8 -102 11 21 31 41 51 61 71 81 91 101
   imp:h,n,p=1  $Steel Shielding
103 0      102      imp:h,n,p=0  $Vaccume surroundings

c Surface cards.
c RCC cells
1 RCC 0 0 0 0 0 36.000000 0.5
c
2 RCC 0 0 0.000000 0 0 9.000000 4.000000
3 RCC 0 0 0.000000 0 0 9.000000 8.000000
4 RCC 0 0 0.000000 0 0 9.000000 12.000000
5 RCC 0 0 0.000000 0 0 9.000000 16.000000
6 RCC 0 0 0.000000 0 0 9.000000 20.000000
7 RCC 0 0 0.000000 0 0 9.000000 24.000000
8 RCC 0 0 0.000000 0 0 9.000000 28.000000
9 RCC 0 0 0.000000 0 0 9.000000 32.000000
10 RCC 0 0 0.000000 0 0 9.000000 36.000000
11 RCC 0 0 0.000000 0 0 9.000000 40.000000
c
12 RCC 0 0 9.000000 0 0 9.000000 4.000000
13 RCC 0 0 9.000000 0 0 9.000000 8.000000
14 RCC 0 0 9.000000 0 0 9.000000 12.000000
15 RCC 0 0 9.000000 0 0 9.000000 16.000000
16 RCC 0 0 9.000000 0 0 9.000000 20.000000
17 RCC 0 0 9.000000 0 0 9.000000 24.000000
18 RCC 0 0 9.000000 0 0 9.000000 28.000000
19 RCC 0 0 9.000000 0 0 9.000000 32.000000
20 RCC 0 0 9.000000 0 0 9.000000 36.000000
21 RCC 0 0 9.000000 0 0 9.000000 40.000000
c
22 RCC 0 0 18.000000 0 0 9.000000 4.000000
23 RCC 0 0 18.000000 0 0 9.000000 8.000000
24 RCC 0 0 18.000000 0 0 9.000000 12.000000
25 RCC 0 0 18.000000 0 0 9.000000 16.000000
26 RCC 0 0 18.000000 0 0 9.000000 20.000000
27 RCC 0 0 18.000000 0 0 9.000000 24.000000
28 RCC 0 0 18.000000 0 0 9.000000 28.000000
29 RCC 0 0 18.000000 0 0 9.000000 32.000000
```

30 RCC 0 0 18.000000 0 0 9.000000 36.000000
31 RCC 0 0 18.000000 0 0 9.000000 40.000000
c
32 RCC 0 0 27.000000 0 0 9.000000 4.000000
33 RCC 0 0 27.000000 0 0 9.000000 8.000000
34 RCC 0 0 27.000000 0 0 9.000000 12.000000
35 RCC 0 0 27.000000 0 0 9.000000 16.000000
36 RCC 0 0 27.000000 0 0 9.000000 20.000000
37 RCC 0 0 27.000000 0 0 9.000000 24.000000
38 RCC 0 0 27.000000 0 0 9.000000 28.000000
39 RCC 0 0 27.000000 0 0 9.000000 32.000000
40 RCC 0 0 27.000000 0 0 9.000000 36.000000
41 RCC 0 0 27.000000 0 0 9.000000 40.000000
c
42 RCC 0 0 36.000000 0 0 9.000000 4.000000
43 RCC 0 0 36.000000 0 0 9.000000 8.000000
44 RCC 0 0 36.000000 0 0 9.000000 12.000000
45 RCC 0 0 36.000000 0 0 9.000000 16.000000
46 RCC 0 0 36.000000 0 0 9.000000 20.000000
47 RCC 0 0 36.000000 0 0 9.000000 24.000000
48 RCC 0 0 36.000000 0 0 9.000000 28.000000
49 RCC 0 0 36.000000 0 0 9.000000 32.000000
50 RCC 0 0 36.000000 0 0 9.000000 36.000000
51 RCC 0 0 36.000000 0 0 9.000000 40.000000
c
52 RCC 0 0 45.000000 0 0 9.000000 4.000000
53 RCC 0 0 45.000000 0 0 9.000000 8.000000
54 RCC 0 0 45.000000 0 0 9.000000 12.000000
55 RCC 0 0 45.000000 0 0 9.000000 16.000000
56 RCC 0 0 45.000000 0 0 9.000000 20.000000
57 RCC 0 0 45.000000 0 0 9.000000 24.000000
58 RCC 0 0 45.000000 0 0 9.000000 28.000000
59 RCC 0 0 45.000000 0 0 9.000000 32.000000
60 RCC 0 0 45.000000 0 0 9.000000 36.000000
61 RCC 0 0 45.000000 0 0 9.000000 40.000000
c
62 RCC 0 0 54.000000 0 0 9.000000 4.000000
63 RCC 0 0 54.000000 0 0 9.000000 8.000000
64 RCC 0 0 54.000000 0 0 9.000000 12.000000
65 RCC 0 0 54.000000 0 0 9.000000 16.000000
66 RCC 0 0 54.000000 0 0 9.000000 20.000000
67 RCC 0 0 54.000000 0 0 9.000000 24.000000
68 RCC 0 0 54.000000 0 0 9.000000 28.000000
69 RCC 0 0 54.000000 0 0 9.000000 32.000000
70 RCC 0 0 54.000000 0 0 9.000000 36.000000
71 RCC 0 0 54.000000 0 0 9.000000 40.000000
c
72 RCC 0 0 63.000000 0 0 9.000000 4.000000
73 RCC 0 0 63.000000 0 0 9.000000 8.000000
74 RCC 0 0 63.000000 0 0 9.000000 12.000000
75 RCC 0 0 63.000000 0 0 9.000000 16.000000
76 RCC 0 0 63.000000 0 0 9.000000 20.000000
77 RCC 0 0 63.000000 0 0 9.000000 24.000000
78 RCC 0 0 63.000000 0 0 9.000000 28.000000
79 RCC 0 0 63.000000 0 0 9.000000 32.000000
80 RCC 0 0 63.000000 0 0 9.000000 36.000000

```
81 RCC 0 0 63.000000 0 0 9.000000 40.000000
c
82 RCC 0 0 72.000000 0 0 9.000000 4.000000
83 RCC 0 0 72.000000 0 0 9.000000 8.000000
84 RCC 0 0 72.000000 0 0 9.000000 12.000000
85 RCC 0 0 72.000000 0 0 9.000000 16.000000
86 RCC 0 0 72.000000 0 0 9.000000 20.000000
87 RCC 0 0 72.000000 0 0 9.000000 24.000000
88 RCC 0 0 72.000000 0 0 9.000000 28.000000
89 RCC 0 0 72.000000 0 0 9.000000 32.000000
90 RCC 0 0 72.000000 0 0 9.000000 36.000000
91 RCC 0 0 72.000000 0 0 9.000000 40.000000
c
92 RCC 0 0 81.000000 0 0 9.000000 4.000000
93 RCC 0 0 81.000000 0 0 9.000000 8.000000
94 RCC 0 0 81.000000 0 0 9.000000 12.000000
95 RCC 0 0 81.000000 0 0 9.000000 16.000000
96 RCC 0 0 81.000000 0 0 9.000000 20.000000
97 RCC 0 0 81.000000 0 0 9.000000 24.000000
98 RCC 0 0 81.000000 0 0 9.000000 28.000000
99 RCC 0 0 81.000000 0 0 9.000000 32.000000
100 RCC 0 0 81.000000 0 0 9.000000 36.000000
101 RCC 0 0 81.000000 0 0 9.000000 40.000000
c
102 RCC 0 0 -10.000000 0 0 110.000000 60.000000

c Data cards
m1 82206 -24.44 82207 -22.94 82208 -52.647
m2 92235 -0.000000 92238 -0.000000 94239 -0.121118 93237 -0.096796 95241
    -0.164797 96243 -0.137331 1001 -0.050121 8016 -0.429836
m3 26000 -0.900000 24000 -0.100000
c LINAC beam, Assumed as a circular disc source
sdef par=9 pos 0 0 0 axs 0 0 1 rad d2 vec 0 0 1 dir 1 erg=1000.000000
si2 0.48
sp2 -21 1
c Tallies
cut:e j 6.7
cut:p j 6.7
mode h n p
phys:n 2010
phys:h 2010
phys:p j 1
totnu
f4:n 2 3 4 5 6 7 8 9 10 11
    12 13 14 15 16 17 18 19 20 21
    22 23 24 25 26 27 28 29 30 31
    32 33 34 35 36 37 38 39 40 41
    42 43 44 45 46 47 48 49 50 51
    52 53 54 55 56 57 58 59 60 61
    62 63 64 65 66 67 68 69 70 71
    72 73 74 75 76 77 78 79 80 81
    82 83 84 85 86 87 88 89 90 91
    92 93 94 95 96 97 98 99 100 101
   e4 0.000001 0.01 1000.000000
print
ctme 1200
```

RFERENCES

1. Francesco Venneri. Disposition of Nuclear Wastes Using Subcritical Accelerator-Driven Systems. The Uranium Institute 24th Annual Symposium. London, 8-10, September 1999.
2. A Roadmap for Developing Accelerator Transmutation of Waste (ATW) Technology. DOE/RW-0519, U.S. Department of Energy. 1999.
3. EUROPEAN TECHNICAL WORKING GROUP ON ADS, A European Roadmap for Developing Accelerator-Driven System (ADS) for Nuclear Waste Incineration. ENEA .2001.
4. Accelerator-Driven Systems: Energy Generation and Transmutation of Nuclear Waste. Status Report, IAEATECDOC-985, International Atomic Energy Agency. 1997.
5. Gregory J. Van Tuyle. Nuclear Applications of Accelerator-Driven Spallation Targets. Nuclear Technology, vol. 122, 330-354. 1998.
6. Yonghee Kim et al. Characterization of a Source Importance Function in an Accelerator-Driven System. Nuclear Science and Engineering, Vol.144, 227-241. 2003.
7. R. Soule et al. Neutronic Studies in Support of Accelerator-Driven Systems: The MUSE Experiments in the MASURCA Facility. Nuclear Science and Engineering, vol. 148, 124-152. 2004.

17. F. Jallu et al. Photoneutron Production in Tungsten, Praseodymium, Copper and Beryllium by Using High Energy Electron Linear Accelerator. Nuclear Instruments and Methods in Physics Research B 155, 373-381. 1999.

18. G. Aliberti et at. Impact of Nuclear Data Uncertainties on Transmutation of Actinides in Accelerator-Driven Assemblies. Nuclear Science and Engineering, Vol. 148, 13-50. 2004.

19. Alexander V. Vornkov et al. A Second-Order Finite Volume Discretization of the Time-Dependent Transport Equation on Arbitrary Quadrilaterals in R-Z Geometry. Nuclear Science and Engineering, Vol. 148, 186-194. 2004.

20. F. Venneri et al. Disposition of Nuclear Waste Using Subcritical Accelerator-Driven Systems: Technology Choices and Implementation Scenarios. Nuclear Technology, Vol. 132, 15-28. Oct. 2000.

21. S. Massara et al. Dynamics of Critical Dedicated Cores for Minor Actinide Transmutation. Nuclear Technology, Vol. 149, 150-174. Feb. 2004.

22. Micah D. Lowenthal. Transmutation in the Nuclear Fuel Cycle: Approaches and Impacts. Nuclear Technology, Vol. 138, 284-299. June 2002.

23. S. B. Sadineni, W. G. Culbreth, et al. Benchmarking Photoneutron Predictions from MCNPX. ANS 2004 Annual Meeting - 50th Anniversary, Pittsburgh, USA. June 13-17, 2004.

24. S. B. Sadineni, W. G. Culbreth et al. Photoneutron Production for Accelerator-Driven Subcritical Systems. ANS 2004 Annual Meeting - 50th Anniversary, Pittsburgh, USA. June 13-17, 2004.

25. Mohamed A. Reda, Frank Harmon and S. B. Sadineni. A Photoneutron Source for a Subcritical Nuclear Reactor Program. 17th International Conference on the Application of Accelerators in Research and Industry, CAARI 2002, University of North Texas Department of Physics Denton, Texas, USA. Nov.12th-16^{th,} 2002.

26. Mohamed A. Reda, Frank Harmon and S. B. Sadineni. A Photoneutron Source for Bulk Material Studies. 2003 Denver X-Ray Conference. Aug. 4-8, 2003.

27. Mohamed A. Reda, Frank Harmon and S. B. Sadineni. Properties of Photoneutron Sources for Accelerator-Driven Subcritical Systems. 2003 ANS Annual Meeting, The Nuclear Technology Expansion – Unlimited Opportunities, San Diego, USA. June 1-5, 2003.

28. Shunsuke Ishimoto et al. Neutronics Study on Accelerator Driven Subcritical Systems with Thorium-Based Fuel for Comparison between Solid and Molten-Salt Fuels. Nuclear Technology, Vol. 138, 300-312. June 2002.

29. Marko Maucec et al. Criticality Analysis of the Multiplying Material Inside the Chernobyl Sarcophagus. Nuclear Technology, Vol. 122. June 1998.

30. Charlton et al. Neutronic Design Studies for the Spallation Neutron Source. Nuclear Technology, vol. 132, 94-101. Oct. 2000.

31. Masaki Saito et al. Transmutation of Elemental Cesium by a Fusion Neutron Source. Nuclear Technology, Vol. 133, 229-240. Feb. 2001.

32. Won Sik Yang et al. Blanket Design studies of a Lead-Bismuth Eutectic-Cooled Accelerator Transmutation of Waste System. Nuclear Technology, Vol. 135, 162-181. Aug. 2001.

33. W. M. Stacey et al. A Subcritical, Gas-Cooled Fast Transmutation Reactor with a Fusion Neutron Source. Nuclear Technology, Vol. 150, 162-188. May 2005.

34. Sukesh K. Aghara et al., "Feasibility Study of a Proliferation-Resistant Fuel form for Plutonium Recycling," Nuclear Technology, Vol. 137, 1-9, (Jan. 2002).

35. Kenji Nishihara et al. Transmutation of ^{129}I Using an Accelerator-Driven System. Nuclear Technology, Vol. 137, 47-59. Jan. 2002.

36. John R. Lamarsh and Anthony J. Baratta. Introduction to Nuclear Engineering. Prentice Hall, Third edition. 2001.

37. James J. Duderstadt and Louis J. Hamilton. Nuclear Reactor Analysis. John Wiley & Sons. 1976.

38. Richard Stephenson. Introduction to Nuclear Engineering. McGraw-Hill, Second edition. 1958.

39. Richard Wolfson. Nuclear Choices A Citizen's Guide to Nuclear Technology. Massachusetts Institute of Technology. 1991.

40. Harry Henderson. Contemporary World Issues Nuclear Power. ABC-CLIO. 2000.

41. Josef R. Parrington et al. Nuclides and Isotopes. General Electric Co. and KAPL, Fifteenth edition. 1996.

42. J. Roseenblatt. Particle Acceleration. Methuen & Co. Ltd. 1968.

43. Laurie S. Waters. MCNPX™ user's manual. Version 2.1.5.

44. MCNP4B2™ User's Manual. Jan. 1997.

45. Suresh B Sadineni. Benchmarking Photoneutron Production of MCNPX with Experimental Results. Masters Thesis, University of Nevada Las Vegas. 2002.

46. K. Shankar Rao. Numerical Methods for Scientists and Engineers. Prentice-Hall India, Second Edition. 2004.

47. A. R. Foster and R. L. Wright, Jr. Basic Nuclear Engineering. Fourth Edition. 1983.

48. W. J. Minkowycz et al. Handbook of Numerical Heat Transfer. John Wiley & Sons. 1988.

49. E. Balagurusamy. Programming in ANSI C. Tata McGraw-Hill, Second Edition. 1997.

50. R. Johnsonbaugh, M. Kalin. C for Scientists and Engineers. Prentice-Hall. 1997.

51. Glen E. Myers. Analytical Methods in Conduction Heat Transfer. AMCHT Publications, Second Edition. 1998.

52. D. Beller et al., The U.S. AFCI Reactor-Accelerator Coupling Experiments (RACE) Project. International Conference on Accelerator Applications 2005 (AccApp05), Venice, Italy, August 31, 2005.

53. J. Chen et al. ISU Accelerator-Driven Sub-critical System Characterization. 2005 Winter Meeting of the American Nuclear Society, Nov. 13-17, 2005.

54. V. V. Kulik et al. Dynamic Analysis of Space-Time Effects in the ISU RACE Configuration. International Conference on Accelerator Applications 2005 (AccApp05), Venice, Italy, August 31, 2005.

55. F. Carminati et al. An Energy Amplifier for Cleaner and Inexhaustible Nuclear Energy Production Driven by a Particle Beam Accelerator. CERN/AT/93-47 (ET), CERN. 1993.

56. Lewis W. Report AECL-969, Atomic Energy of Canada Ltd. 1953.

57. Steinberg M (ed) et al. Proceedings of Information Meeting on Accelerator-Breeding. Report Conference 770107, Brookhaven National Laboratory, 1977.

58. Grand P et al. Conceptual Design and Economical Analysis of Light Water Reactor Fuel Enrichment/Regerator. BNL 50838 (UC-80, TID-4500), Brookhaven National Laboratory, 1978.

VITA

Graduate College
University of Nevada, Las Vegas

Suresh Babu Sadineni

Local Address:
4386 Escondido Street
Apartment # 104
Las Vegas, Nevada 89119
USA

Home Address:
Koppole (P.O)
Ongole (M.D)
Prakasam (D.T), A.P.
INDIA 523 287

Degrees:
Bachelor of Engineering, 2000
Andhra University College of Engineering, Visakhapatnam, INDIA

Master of Science, 2002
University of Nevada, Las Vegas, USA

Publications:

1. Suresh B. Sadineni, William G. Culbreth and Frank Harmon, "*Benchmarking Photo-Neutron Predictions from MCNPX.*" ANS 2004 Annual Meeting – 50th Anniversary, Pittsburgh (USA), 13-17 June (2004).

2. Suresh B. Sadineni, William G. Culbreth and Frank Harmon, "*Photo-Neutron Production for Accelerator-Driven Sub critical Systems.*" ANS 2004 Annual Meeting – 50th Anniversary, Pittsburgh (USA), 13-17 June (2004).

3. M. A. Reda, J. F. Harmon, and S. B. Sadineni , "*Photo-Neutron Source for a Sub-Critical Nuclear Reactor Program.*" 17th International Conference on the

Application of Accelerators Research and Industry, Denton, Texas (USA), 12-16 November (2002).

4. M. A. Reda, J. F. Harmon, and S. B. Sadineni , *"Properties of Photo-Neutron Sources for Accelerator Driven Sub-Critical Systems."* 2003 ANS Annual Meeting, "The Nuclear Technology Expansion-Unlimited Opportunities", San Diego, (USA), 1-5 June (2003).

5. M. A. Reda, J. F. Harmon, and S. B. Sadineni , *"A Photo-Neutron Source for Bulk Material Studies."* 2003 Denver X-Ray Conference, 4-8 August (2003).

Dissertation Examination Committee:

Chair Person, Dr. William Culbreth Ph. D.
Committee Member, Dr. Robert F. Boehm Ph. D.
Committee Member, Dr. Ajit K. Roy Ph. D.
Adjunct Faculty Member, Dr. Frank Harmon Ph. D.
Graduate College Representative, Dr. Sahjendra Singh ph. D.

8. Ed Arthur. Advanced Accelerator Application: Addressing Nuclear Issues. 2001 Science Day, Los Alamos National Laboratory, Feb.13, 2001.

9. S. Dulla et al. Transport Effects for Source-Oscillated Problems in Subcritical Systems. Nuclear Science and Engineering, Vol. 148, 89-102. 2004.

10. C. D. Bowman. Nuclear Energy Generation and Waste Transmutation using Accelerator-Driven Intense Thermal Neutron Source. Nuclear Instruments and Methods in Physics Research A320 336-337, North-Holland. 1992.

11. C. D. Bowman. Accelerator-Driven Systems for Nuclear Waste Transmutation. Annu. Rev. Nucl. Part. Sci., 48: 505-56. 1998.

12. S. M. Seltzer and M. J. Berger. Photoneutron Production in Thick Targets. Physical review C, Vol. 7, number 2. Feb. 1973.

13. William P. Swanson. Improved Calculation of Photoneutron Yields Released by Incident Electrons. Health Physics Vol. 37, pp 347-358. Sept.1978.

14. G. P. Lawrence. High Power Linear Accelerators for Tritium Production and Transmutation of Nuclear Waste. Nuclear Instruments and Methods in Physics Research B56/57, 1991, 1000-1004, North Holland. 1991.

15. J. U. Knebel et al. Thermalhydraulic and Corrosion Challenges for the Target Module of an Accelerator-Driven System (ADS). Third International Topical meeting on Nuclear Application of Accelerator Technology AccApp'99, Long Beach CA, 14-18, pp 367-376. November 1999.

16. J. U. Knebel et al. Research on Transmutation and Accelerator-Driven Systems. Internationale Zeitschrift Fur Kernenergie atw Jg. 2000. Heft 6, pp. 350-358. June 2000.