COMPUTER REASONING ABOUT NUCLEAR PHYSICS PROCESSES

by

Sri Hartati

TR96-107, April 1996

This is an unaltered version of the author's
Ph.D. Thesis

Faculty of Computer Science
University of New Brunswick
Fredericton, N.B. E3B 5A3
Canada

Phone: (506) 453-4566
Fax: (506) 453-3566
E-mail: fcs@unb.ca
www: http://www.cs.unb.ca

Computer Reasoning About Nuclear Physics Processes

by

Sri Hartati

Dra.(Electronics) — Gajah Mada University

M.Sc.(CS) — University of New Brunswick

A Thesis Submitted in Partial Fulfillment of

the Requirements for the Degree of

Doctor of Philosophy

in the

Faculty of Computer Science

This thesis is accepted.

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

March, 1996

Abstract

Nuclear Process Theory (NPT) is a new process theory which models nuclear physics processes using a formal grammar. This, in turn, allows one to write computer programs which simulate nuclear process interactions. Knowledge of the nuclear physics process is expressed in terms of a basic nuclear physics model and an aggregate effects model. The basic nuclear physics model has the capability of expressing knowledge of nuclear physics processes for different types of reactions. The products of reactions are expressed in either deterministic or probabilistic values. The aggregate effects model coordinates basic nuclear physics processes expressed in the basic nuclear physics model.

NPT has been tested by using it to represent several nuclear physics processes, including fission, radioactive capture, beta negative and positive decay, that occur in a homogeneous reactor core. A simulator called NPTsim was written in C to demonstrate the principles of NPT representation.

Three examples of running 20 second simulations have been carried out using NPTsim. The three experiments used differing amounts of ^{235}U and ^{238}U corresponding to pure, natural and enriched Uranium. The experimental results are very closed to expected theoritical results.

Unique approaches to symbolically representing the state space and probabilistic space equations required for nuclear physics processes are presented. In addition, a unique explanation mechanism for how products of nuclear physics processes arise is enabled by the NPT approach. This mechanism is illustrated in the thesis.

Contents

Abstract ii

List of Tables vii

List of Figures viii

Acknowledgements x

Dedication xii

Nomenclature xiii

1 Introduction 1

1.1 Background 2

1.2 Objectives of the thesis 5

1.3 Outline of the thesis 6

2 Reasoning about physical systems 7

2.1 Qualitative Process Theory (QPT) 8

2.1.1 Process 9

2.1.2 Basic deductions 10

2.2 Qualitative Simulation (QSim) 13

2.2.1 Derivation of behaviour 13

2.3 Envision 16
2.3.1 Device structure 16
2.3.2 Structure to function 17
2.3.3 Deriving behavior 18
2.4 Dimensional Analysis 19
2.5 Hybrid Phenomena Theory (HPT) 22
2.5.1 Components of HPT 22
2.5.2 Influences in HPT 25
2.5.3 Subsumption 27
2.6 Comparison 28

3 Nuclear Processes **32**
3.1 Behaviour of Nuclear Physics Processes 36
3.2 Neutron Cycle 37

4 Nuclear Process Theory (NPT) **42**
4.1 Knowledge Level 42
4.1.1 Basic Nuclear Physics Model (BNPM) 43
4.1.2 Aggregate Effects Model (AEM) 49
4.2 Symbolic Level 56
4.3 Quantitative Level 58
4.4 Influences in NPT 60
4.5 Deriving State Space and Probabilistic Space Equations 65
4.5.1 Algebraic Influences 65
4.5.2 Aggregate Influences 66
4.5.3 Cumulative Influences 66
4.5.4 Decay Influences 67
4.5.5 Distribution Influences 68

5 Implementation **71**

5.1 Design and architecture of a nuclear process simulation tool 71

5.2 NPT language 79

6 Evaluation **84**

6.1 Experiments 85

6.1.1 State space and probabilistic space equations of the first cycle for experiment 2 90

6.1.2 State space and probabilistic space equations of the second cycle for experiment 2 100

6.2 Comparison to Expected Results 103

6.3 Queries, Histories and Changes parameters 105

6.4 Worst Case Time Analysis 108

6.5 Generalization 110

6.5.1 Environmental Modeling 110

6.5.2 Chemical processes 110

6.5.3 Fusion 111

6.5.4 Nuclear Medicine 112

6.5.5 Communication Networks 114

6.5.6 Software Validation 114

7 Conclusions and Future Work **115**

7.1 Conclusions 115

7.2 Suggestions for future work 116

Bibliography **118**

Appendices

I Equations Used in NPT **122**

I.1 Fission processes in NPT 122
I.1.1 Macroscopic cross section 123
I.1.2 Fission fraction 125
I.1.3 Thermal utilization factor 127
I.1.4 Thermal fission factor 128
I.1.5 Multiplication factor 129
I.1.6 Fissions per cycle 130
I.1.7 Fission products 131
I.1.8 Fuel Consumption 140
I.2 Beta negative decay in NPT 141
I.3 Beta positive decay in NPT 144
I.4 Radioactive capture in NPT 145

II NPTsim Input descriptions for Experiment 2 **147**

III Formal Grammar of the NPT language **149**

IV Set of Active Processes at the End of the Second Cycle **156**

V State and Probabilistic Space Equations at the End of the Second Cycle **198**

VI Probabilistic space equations of the sorted isobars **243**

VII Final results of Experiment 2 **270**

List of Tables

2.1 Comparison of physics reasoning systems. 31

6.1 Portion of distr_A. 97

6.2 Portion of pdf_dZ. 97

I.1 Probability of fission fragments having a certain mass number for activating nuclide U235. 133

I.2 Probability per unit interval of energy of neutron products calculated from [13]. 137

I.3 Probability per unit interval of energy of gamma products. 138

List of Figures

2.1 Physical process definition for fluid flow from [11]. 11
2.2 Model components in HPT. 24
2.3 A pan with water on hotplate. 28
3.1 Heavy elements of interest for nuclear reactors from [21]. 34
3.2 A simplified diagram for nuclear physics processes. 35
3.3 Neutron cycle in NPT. 39
4.1 Model hierarchy in NPT. 43
4.2 Basic nuclear physics model of fission. 44
4.3 Basic nuclear physics model of radioactive capture. 45
4.4 Basic nuclear physics model of beta positive decay. 45
4.5 Basic nuclear physics model of beta negative decay. 46
4.6 Aggregate effects model of fission. 51
4.7 Continuation of aggregate effects model of fission. 52
4.8 Aggregate effects model of radioactive capture. 53
4.9 Aggregate effects model of beta negative decay. 54
4.10 Aggregate effects model for beta positive decay. 55
4.11 Example of a set of conceivable processes at the end of the first cycle. 58
4.12 Active processes at the end of the first cycle of the experiment. 59
5.1 Components of a nuclear process simulation tool. 72
5.2 High-level activities of simulating nuclear processes in a cycle. 73
5.3 High-level description of simulation using the NPT model. 77

5.4 Functional model of NPTsim. 80
5.5 Basic organization of the NPT language processor. 81
6.1 Time line used in NPT. 85
6.2 Active processes at the beginning of the first cycle of the Experiment. 86
6.3 State and probabilistic space equations at the end of the first cycle of Experiment 2. 87
6.4 Continuation of state and probabilistic space equations at the end of first cycle of Experiment 2. 88
6.5 Portion of active processes at the end of second cycle of Experiment 2. 89
6.6 Isobar line. 96
6.7 A portion of probalistic space equations for fission products at t_1 for experiment no. 2. 98
6.8 Continuation of a portion of probalistic space equations for fission products at t_1 for experiment no. 2. 99
6.9 Portion of the state and probabilistic space equations at time t_2 Experiment 2. 102
6.10 Example of history and query. 106
6.11 Continuation of an example of history and query. 107
II.1 NPTsim input descriptions of experiment no.2. 148

Acknowledgements

I wish to express my sincere thanks and appreciation to my supervisor Dr. Bradford G. Nickerson for his useful advice, guidance, constructive criticism given throughout the extent of this research, his patience and for suggesting a number of improvements in my use of English grammar.

I would express my sincere thanks to Dr. George Demille for his useful discussions and for his guidance for better understanding of nuclear processes.

I wish to thank committee members, Dr. Bruce Spencer and Dr. David Fellows, for their suggestions and comments generously expressed at various stages of conducting my research.

I greatly appreciate and thank the Government of Indonesia for the financial support and the opportunity to study at the University of New Brunswick, Fredericton, Canada as well as to bring my family to Fredericton.

I also would like to express my sincere thanks to the staff and faculty of the UNB Faculty of Computer Science for giving me very useful training and guidance through the courses I took, and also giving me an opportunity to carry out my research. A special thanks goes to Kirby Ward for his suggestion of solutions when I have had difficulties on systems in the AI Lab.

I would like to express my deepest feeling of gratitude, in particular, to:

- Devica, my beloved daughter, who cheers up and refreshes me every day after doing some work related to my research on campus.

- All moslem sisters who helped me to make my stay at Fredericton more enjoyable and make myself stronger in faith.
- My beloved mother Wiryodaryanto for her wonderful patience in taking care of my daughter and her moral support during my difficult years of study. She always encouraged me to be steadfast pursuing knowledge.
- My husband Agus, for his wonderful support and understanding during the difficult hours of my work, and for his encouragement.
- The Sosrowidarsono's, my parents in law, for their moral support during my study.

Dedication

This thesis is dedicated to my beloved *mother* who first introduced me to the *Qur'an* particularly chapter 112, which means

In the name of God, Most Gracious, Most Merciful

1. *Say: He is God, The One and Only;*
2. *God, the Eternal, Absolute;*
3. *He begetteth not, nor is He begotten;*
4. *And there is none like unto Him.*

Nomenclature

A	:	mass number
A_c	:	mass number of a compound nucleus
C^T_{bnd}	:	nuclide consumption due to the beta negative decay process
C^j_{bnd}	:	amount of nuclides of type j consumed for beta negative decay processes
C^j_{rc}	:	nuclide consumption due to the radioactive capture process, as a result of interactions between neutrons and nuclides of type j
E_P	:	energy released from product nuclide(s)
E_b	:	energy released from product particle(s)
F_1	:	fission products 1
F_2	:	fission products 2
F^j_a	:	a fraction of absorption of neutrons for each type of nuclide
F_f	:	fraction of thermal neutrons participating in the fission process
F^j_f	:	fraction of neutrons which interact with nuclides of type j through fission
F^j_g	:	fraction of neutrons which interact with nuclides of type j through radioactive capture
N	:	neutron number
NA	:	Avogadro's number
N_i	:	density of nuclides of type i
P_{NLf}	:	fast nonleakage probability
P_{NLth}	:	thermal nonleakage probability
P^R_{bnd}	:	resulting nuclide due to a beta negative decay process

Q	:	a nuclear physics system
P	:	product nuclide(s)
$P(Z,A).amount$	:	number of resulting nuclides P
R_{c_g}	:	number of fissions in a present cycle as a result of fissioning certain types of nuclides
R^j_{rc}	:	resulting nuclides due to radioactive capture process, as a result of interactions between neutrons and nuclides of type j
SP	:	a collection of state space and probabilistic space equations
T	:	target nuclide
X	:	total number of fission neutrons in a previous generation
Z	:	atomic number
Z_c	:	atomic number of a compound nucleus
Σ_a^{cl}	:	macroscopic absorption cross section for cladding
Σ_a^{fuel}	:	a macroscopic absorption cross section
Σ_f^{fuel}	:	macroscopic fission cross section of a mixture of n substances
Σ_a^{mod}	:	macroscopic absorption cross section for moderator
δ_Z	:	deviation of product nuclide
ϵ	:	fast fission factor
η	:	thermal fission factor
ν	:	average number of neutrons released per fission
σ_f^i	:	fission cross section of nuclides of type i
σ_g^i	:	capture cross section of nuclides of type i
a	:	incident particle
b	:	product particle(s)
$distr\_neutron^j$	:	distribution of fission neutrons for nuclide of type j

ei	:	energy interval
f	:	thermal utilization factor
k	:	a multiplication factor
n_{g-1}	:	neutrons in a previous cycle
n_g	:	neutrons in a current cycle
n_{th}^{fuel}	:	fraction of thermal neutrons absorbed in fuel
p	:	resonance escape probability
$pdf\_neutron$	:	Poisson distribution controlling a probability of producing 0-8 neutrons
pn^j	:	total number of fission neutrons generated by nuclides of type j
q	:	a nuclear physics process

Chapter 1

Introduction

Computer reasoning about the physical world is an important topic of investigation. Traditionally, computer programs or "computer codes" are written in Fortran, C or other computer languages that support large scale computations. Recently, work in Artificial Intelligence has attempted to model and represent physical processes in such a way that computer programs are much more "human-like" in the way they are constructed, e.g. Maple, MODSIM, and G2. This allows them to be used in a much more natural way for explaining how particular solutions were obtained. Complex problem solutions can be easily formulated.

This thesis presents a formal representation system for nuclear physics processes. It introduces the Nuclear Process Theory, NPT, a framework integrating symbolic descriptions of nuclear physics processes with state space and probabilistic space equations along with the products of nuclear physics processes. The key idea explored here is how nuclear physics processes can be represented to model nuclear physics interactions both quantitatively and qualitatively which, in turn, enables computers to reason symbolically about nuclear physics processes. It is emphasized that the purpose of the approach described here is to enable a formal description of what is already known as nuclear physics processes.

NPT models nuclear physics processes using a formal grammar. This, in turn,

allows one to write computer programs which simulate nuclear process interactions. Knowledge of the nuclear physics processes is expressed in terms of a basic nuclear physics model and an aggregate effects model.

1.1 Background

Qualitative physics is concerned with representing and reasoning about the physical world. Both qualitative physics and physics attempt to characterize the physical world by formalizing knowledge about it in some language. Physics attempts to elaborate on new theories, but does not bother to codify the knowledge for direct computer manipulation. To develop machines capable of reasoning about physical systems, the knowledge must be represented in a fashion suitable for machine manipulation.

Reasoning about physical systems aims to expose the underlying mechanisms and make them sufficiently explicit, so that they can be directly reasoned with and about. Reasoning about physical systems can give causal explanations of a change in one variable with respect to others. It can reason about when a process starts and stops, and can give causal explanations when the implicit assumptions under which it was written are violated. The underlying and explicit knowledge enables us to develop machines capable of reasoning about physical systems. Artificial Intelligence (AI) research on reasoning about physical systems is important for several reasons, e.g. [34]

- It allows robots to predict the effect of their actions on a dynamic world.
- It leads to powerful tools for automated design, diagnosis, and monitoring of complex systems such as electronic circuits or chemical plants.
- It enables the construction of algorithms that generate causal descriptions explaining how physical systems work; these algorithms could be used in intelligent tutoring systems.

Reasoning about physical systems is one of the fields of AI where attempts have been made to describe and analyze the behaviour of physical systems. Qualitative reasoning about physical systems has attracted a growing interest since the late 70's. Research in qualitative reasoning was originally motivated by the fact that human beings function well in their physical surroundings without resorting to quantitative computations. The first work was an exploration of how qualitative reasoning could be used to guide the construction and solution of equations in simple motion problems as well as answering simple questions directly [5]. Qualitative reasoning has from the beginning been concerned with reasoning about qualitative properties. A number of different qualitative reasoning tasks that have inspired work on several styles of reasoning are [8]

Simulation: Determining a likely course of future behaviour starting with a structural description of some device or system, and some initial conditions.

Envisionment: Starting with a structural description, determine all possible behavioural sequences.

Diagnosis: Determining the cause of a change in the underlying structure of systems which previously behaved correctly.

Verification: Designed systems start with a behaviour specification to achieve a system function. Any number of structures may be used to try to implement such a specification. The problem of verification is to ascertain that a particular implementation structure has a composite behaviour which matches the desired behaviour specification.

Deducing functionality: Extracting functional descriptions from structural and behavioural descriptions to identify the functions of components in the system.

A variety of productive approaches to qualitative reasoning frequently differ in emphasis and content, and can seem incompatible [2,3,11,12]. These approaches focus

on different methods for reasoning about different types of physical systems.

Qualitative representation is often appropriate for the system that is lacking complete numerical information. It provides a theoretical framework for understanding the behaviour of a physical system, e.g. a qualitative expectation of the system's behaviour over time.

Among developed methods in qualitative reasoning, only Forbus' method [11] introduced the notion of a process, where physical interactions are described through properties of processes rather than properties of devices.

Although the approaches mentioned above lack quantitative information, they have provided valuable insight and methods for analyzing physical systems. Most application areas in science and engineering also require some form of quantitative analysis. Several attempts on integrating qualitative knowledge and quantitative knowledge for reasoning about physical system have been developed [10,24,36]. These approaches produce more accurate predictions and causal explanations of behaviour of physical systems. Besides giving numerical descriptions of the behaviour, these approaches describe the behaviour qualitatively, and provide a causal explanation of behaviour.

Woods [36] introduces a Hybrid Phenomena Theory which also incorporates the notion of physical processes for describing the behaviour of physical systems. Woods introduced a combined qualitative and quantitative approach that can have a quantitative interpretation. His approach introduced parametric state space models which can represent parameters that physicists and engineers normally use and can easily understand.

Research on both qualitative and quantitative reasoning emphasizes development of methodologies for reasoning about physical systems. None of the above approaches have been developed to reason about nuclear physics processes [14]. Nuclear processes are somewhat different from other physical systems, as nuclear physics processes

transform original nuclides to new materials, and produce particles. To develop machines capable of reasoning about nuclear physics processes, the knowledge about physical processes, physical system components and the knowledge about properties of approximately 3000 different types of nuclides [18] must be codified and embodied in the system.

1.2 Objectives of the thesis

The thesis concentrates on developing an approach for computer reasoning about nuclear physics processes. This work has two primary objectives, namely:

1. Devise a theory for defining the nuclear physics processes that can be developed as a language in which to write computer programs which simulate nuclear physics interactions.

2. Investigate the utility of the theory as a tool for simulation of nuclear physics processes, including fission processes, radioactive capture processes, and beta positive and beta negative decay processes.

Our approach, called Nuclear Process Theory (NPT), is addressed at the issues of developing a representation of nuclear physics processes that allows understanding of how nuclear interactions (interactions between reagents) dynamically relate to equations describing the processes. Parameters in these equations can have nondeterministic values. Knowledge of nuclear physics processes is expressed in terms of a basic nuclear physics model and an aggregate effects model. The aggregate effects model coordinates basic nuclear physics processes expressed in the basic nuclear physics model.

1.3 Outline of the thesis

The organization of this thesis is as follows. In chapter 2 a general overview of reasoning about physical systems is given. Several different approaches for reasoning about physical systems are presented and compared.

In Chapter 3 an overview of nuclear processes is given, along with reactor parameters that are needed in the processes.

In Chapter 4, Nuclear Process Theory, a formal knowledge representation system for nuclear physics processes, is described in detail, while the algorithms and the implementation are described in Chapter 5.

In Chapter 6 some examples along with evaluations of the performance of NPT is described, followed by some conclusions reported in Chapter 7. In the final Chapter 7 suggestions for further work are given.

Chapter 2

Reasoning about physical systems

Reasoning about physical systems has become an intensive research area of Artificial Intelligence in recent years. Research on qualitative reasoning has been motivated by efforts to make computers more intelligent and capable of reasoning about physical worlds much as we ourselves, as engineers and scientists, do. Considerable emphasis has been placed on a kind of analysis called qualitative simulation, for example [6, 11,12,22]. Other efforts have been made in the domain of modeling functions and malfunctions of devices, causal processes that use devices (for example [35]), and in the domain of assembling a device [9]. Another focus has been on the problem of comparative analysis. It takes as input a system's behaviour and a perturbation and outputs a description of how and why the behaviour would change as a result of the perturbation [34]. Alternative approaches for reasoning about different types of physical systems are included in [33]. The various approaches in reasoning about physical systems have no uniform notation, which makes it hard for one to recognize the common aspects of the different approaches.

Qualitative reasoning provides valuable insights and methods for analyzing physical systems. With the lack of complete quantitative information, qualitative reasoning can provide an understanding of the qualitative nature of a system's behaviour. For example, in design one may not know the exact value of all parameters in the design,

yet one has to make decisions using this partial information. In this case the partial information can be used by representing it in qualitative form. By using qualitative descriptions of the variables, a description of the working of a device can be obtained. Since qualitative values have less information than numerical values, some chains of qualitative reasoning result in ambiguity. Any ambiguity is due to the weakness of algebraic operations that can be defined on a qualitative value space. For example, addition of a negative qualitative value to a positive qualitative value results in an ambiguous value.

Many engineering applications involving the physical world require quantitative values. Research on reasoning about physical systems that combines quantitative information and qualitative descriptions allows a more precise characterization of system and their behaviours [24,36].

This section reviews some methods in qualitative reasoning about physical systems. Among the methods reviewed below, Forbus' and Woods' methods introduce the notion of processes.

2.1 Qualitative Process Theory (QPT)

A purely qualitative approach developed for analyzing physical systems is Qualitative Process Theory (QPT) [11]. QPT defines a simple notion of physical processes which is useful as a language to write dynamic theories.

The central concept in Forbus' analysis is that of the process. Instead of focusing on what processes take place in the system, Forbus' method focuses on how the processes influence the system parameters and how the processes interact.

A physical system is described by objects and relationships between the objects. Processes are represented by activities that occur in physical systems. The significant properties of a process are its preconditions and its influences.

The preconditions of a process are states that must hold if the process is to be

active. For example, consider the process of heat flow from object A to object B. Sufficient and necessary preconditions for heat flow are that A and B are thermally connected, and that the temperature of A is greater than the temperature of B. The influences of the heat flow are to reduce the temperature of A and to increase the temperature of B.

Besides characterizing processes, QPT describes the connections between parameters. Interactions between parameters are described by the concept of direct and indirect influences. Direct influences are used to describe dynamic interactions whereas indirect influences are used to describe functional relationships.

QPT expresses direct influences between parameters using the notations $I+(k,n)$ and $I-(k,m)$. The first notation is pronounced "k influenced by n positively", and the second is "k influenced by m negatively". The value of the derivative of k is equal to the qualitative sum of the influences of all processes on the parameter. If all signs of the influences are the same, (e.g. -1,0,1 indicating decreasing, unchanging and increasing parameters), then the value of the derivative of the influenced parameter is simply that sign. Since there is no numerical information, ambiguities can arise if the signs of the influences are different.

The relations between parameters that are influenced indirectly by processes are expressed using statements of proportionality. Parameter k is qualitatively proportional to m, denoted as $k \propto_{Q+} m$, means an increase in k will also increase m, while other parameters remain the same. Parameter k is negatively proportional to m, written $k \propto_{Q-} m$, if an increase in k will cause a decrease in m.

2.1.1 Process

A physical process is something that acts through time to change the parameters of objects. The concept of a process is used to capture the effects of dynamic interaction in a system. Take a pan on a hot plate as an example. The mere fact that a container is placed on a hot-plate does not entail that a flow of heat will occur. But if the

temperature in the plate is greater than the temperature in the container, a flow of heat will arise. This will cause the temperature in the container to rise and the temperature in the plate to decrease. The notion of a process allows us to formalize conditions for when a process will exist as well as the consequences of its activity. A process is specified by five parts :

Individuals: description of the objects which the process acts on.

Quantity Conditions: Inequality statements and status assignments which must be true for the process to be active.

Preconditions: Statements other than Quantity Conditions that must be true for the process to be active.

Relations: The relationships between the individuals which hold when the process is active.

Influences: Descriptions of the direct effects of the process.

Processes act between any collection of individuals they match, whenever both Preconditions and Quantitative Conditions are satisfied. This gives rise to Process Instances or PIs. The statements in the Relations and Influences fields hold whenever a process is active. Fig. 2.1 shows the process definition of fluid flow.

2.1.2 Basic deductions

QPT representation allows several deductions as categorized below [11].

Finding possible processes: Given an initial set of objects, find all collections of them which match the specifications of each individual view, with the process definitions. Each collection gives rise to a particular view instance (VI) and process instance (PI). These process instances (PIs) represent potential processes

process fluid-flow
Individuals:
    src a contained-liquid
    dst a contained-liquid
    path a fluid path, Fluid-Connected(src,dst,path)
Precondition:
    Aligned(path)
QuantityConditions:
    A[pressure(src)] > A[pressure(dst)]
Relations:
    Let flow-rate be a parameter
    flow-rate$\propto_{Q+}$ (A[pressure(src)]-A[pressure(dst)])
Influences:
    I+(amount-of(dst),A[flow-rate])
    I-(amount-of(src),A[flow-rate])

Figure 2.1: Physical process definition for fluid flow from [11].

that can occur between sets of individuals in a view. Individual views describe objects (called individuals) and their states.

Determining Activity: A process instance is either active or inactive according to whether or not the particular process it represents is acting between its individuals. A status instance is assigned to each process instance for a system by determining whether or not the preconditions and the parameter conditions are true. This gives rise to the Process and View structures - the collection of the active PIs and the collection of VIs of a system, respectively. The process structure represents what is happening to the individuals in a particular system.

Determining changes: Most of the changes in an individual are represented by the D_s values for its parameters, where D_s is a sign of the derivative. A D_s value of -1 indicates the parameter is decreasing, a value of 1 indicates the parameter is increasing, and a value of 0 indicates that it remains constant. There are two ways for a parameter to change; either it is caused directly by a process

or influenced indirectly by $\propto_Q$. Determining the value of D_s for a parameter is called resolving its influences [11]. If a parameter is directly influenced, resolving that parameter requires adding up the influences. If all influences have the same signs, then the value is simply that sign. Since there is no numerical information, ambiguities can arise. Sometimes an answer can be found by sorting the influences into positive and negative sets, and using inequality information to prove that one set of influences must, taken together, be a larger set than the other set. However there is not always enough information to do this, so direct influences are not always resolvable [11].

Resolving an indirectly influenced parameter involves gathering the statements D_s that specify it as a function of other parameters [11]. In many cases indirect influences cannot be resolved within QPT because of the lack of detailed information about the form of the function. For example, suppose there is a parameter q_0 such that a particular process structure holds, as follows:

$$q_0 \propto_{Q+} q_1 \wedge q_0 \propto_{Q-} q_2$$

where $\wedge$ represents a boolean operation AND. If we also know that $D_s[q_1] = 1$ and $D_s[q_2] = 1$, then $D_s[q_0]$ cannot be determined, since there is not enough information to determine which indirect influence dominates. However, if we had $D_s[q_1] = 1$ and $D_s[q_2] = 0$, then we can conclude that $D_s[q_0] = 1$.

Limit Analysis: Process Structures (PSs) and View Structures (ISs) can change due to changing parameters. Limit Analysis is defined as determining the possible changes in a Process Structure and View Structure. It is carried out by using D_s values and the collection of quantity conditions to determine which parameter conditions can change. Limit Hypotheses are determined to define possible situations for ending a Process Structure.

Several examples illustrating some of the basic deductions possible using QPT are explained in detail in [11]. The examples include fluid flow, heat flow, and motion

processes that involve liquids, gases, and solids.

2.2 Qualitative Simulation (QSim)

Qualitative Simulation (QSIM) was developed by Kuipers [23]. QSIM is a purely mathematical approach, and is very different from QPT. It does not attempt to establish a qualitative model from a description of the system to be analyzed. QSIM's basic objective is to derive a qualitative description of the dynamic behaviour of physical systems. The behaviour of a physical system is described by values of a set of variables as a function of time. Kuipers's approach starts with a set of constraints abstracted from a differential equation, and produces a qualitative behaviour corresponding to any solution to the original equation. Kuipers introduces the concept of a qualitative function and expresses all constraints on the behaviour of the functions. At any time, a function takes on one of two types of values; a landmark, or an interval between two landmarks. Time is described by means of distinguished time-points, which occur whenever a function reaches or leaves a landmark. Time is either equal to a distinguished time-point, or it equals an interval between two such points of time. He also points out that both analytical and qualitative solutions are abstractions of the real behaviour of a physical system.

2.2.1 Derivation of behaviour

The notion of a qualitative state is one of the important concepts of qualitative reasoning. The qualitative state of a variable is the qualitative value of the variable [11]. The qualitative state of a physical system is the combination of the states of all variables describing the system. Kuipers [23] made an extension of the notion of a state by including derivatives on an equal basis with the values of functions. Kuipers thus defines the state and behaviour of a single function in the following way:

State of a QSIM function

Let $L_1 < \cdots < L_n$ be the landmark values of $f : [a, b] \rightarrow R$. For any $t \in [a, b]$, the qualitative state of f at t, $QS(f, t)$ is a pair $< qval, qdir >$ defined as follows:

$$qval = \begin{cases} L_i & \text{if } f(t) = L_i, \text{ a landmark value} \\ (L_i, L_{i+1}) & \text{if } f(t) \in (L_i, L_{i+1}) \end{cases}$$

$$qdir = \begin{cases} inc, & \text{if } f(t) > 0 \\ dec, & \text{if } f(t) < 0 \\ std, & \text{if } f(t) = 0 \end{cases}$$

Qualitative behaviour

The qualitative behaviour of f on $[t_a, t_b]$ is the sequence of the qualitative states of f:

$$QS(f, t_a), QS(f, t_a, t_{a+1}), \cdots, QS(f, t_{b-1}, t_b), QS(f, t_b)$$

alternating between qualitative states at distinguished time points, and qualitative states on intervals between distinguished time points.

A physical system is characterized by a set of variables. QSIM performs an analysis of a system by examining the set of qualitative functions $f_1, \cdots, f_m$ corresponding to the set of characteristic variables. There are two types of qualitative states for this kind of system, either at landmarks or between landmarks. These are specified as indicated below.

$$QS(F, t_i) = [QS(f_1, t_i), \cdots, QS(f_m, t_i)] \tag{2.1}$$

$$QS(F, t_i, t_{i+1}) = [QS(f_1, t_i, t_{i+1}), \cdots, QS(f_m, t_i, t_{i+1})] \tag{2.2}$$

The qualitative behaviour of the system F is specified as the sequence of the qualitative states of F, alternating between landmark and interval values.

$$QS(F, t_0), QS(F, t_0, t_1), QS(F, t_1), \cdots, QS(F, t_n)$$

To derive formally the sequence of states constituting the qualitative description, it is required to identify the possible transitions between states. There are two types

of qualitative state transitions [23]: $P-transitions$, moving from a landmark to an interval, and $I-transitions$, moving from an interval to a landmark. The definitions of these transitions are given below. An $I-transition$ of f is a pair of adjacent qualitative states of f,

$$QS(f, t_{i-1}, t_i) \Rightarrow QS(f, t_i)$$

whose first state is the qualitative state on the interval between distinguished time-points. A $P-transition$ of f is a pair of adjacent qualitative states of f , whose first state is the qualitative state at distinguished time-point.

$$QS(f, t_i) \Rightarrow QS(f, t_i, t_{i+1})$$

The possible state transitions in QSIM can be seen in [14]. The qualitative simulation algorithm works with the following descriptions of a mechanism [23]:

1. Set $f_1, \cdots, f_m$ of symbols representing the functions in the system.
2. Set of constraints applied to the function symbols. Each constraint may have associated corresponding values for its functions.
3. Each function is associated with a totally ordered set of symbols representing landmark values; each function has at least the basic set of landmarks $-\infty, 0, \infty$.
4. Each function may have upper and lower range limits, which are landmark values beyond which the current set of constraints no longer apply.
5. An initial-point symbol, t_0, and qualitative values for each of the f_1 at t_0 are given.

Examples illustrating the mechanisms of QSIM, including the vertical launch and subsequent drop of a ball as well as a simple mechanical spring system are given in [23].

2.3 Envision

This approach was developed by de Kleer and Brown [6]. They presented a framework for modeling the generic behavior of individual components of a device which is based on the notions of qualitative differential equations (confluences) and qualitative state. Their approach was implemented in a computer program called "Envision". Envision first establishes a qualitative model of the system, and then simulates the model in an attempt to derive the behavior of the system. One of the main features of this approach is that it considers a physical system to consist of simpler components. Envision builds a qualitative model of a system from the following inputs: a list of components constituting the system, a topological description of how the components are interconnected, and a library of generic component-models.

2.3.1 Device structure

There are three basic entities in the Envision approach: components, connections and materials. Physical behavior is accomplished by operating on and transporting materials such as water, air, and electrons. Components are constituents that can change the form and characteristics of materials. Connections are simple constituents which transport material from one component to another and cannot change any aspect of the material within them. Some examples of conduits are pipes, wires and cables. Variables are associated with the material flow. Components act on the materials causing the associated variables to change their values.

The components are building blocks. Each component is described by a generic qualitative model. The behavior of a system is derivable from the generic component models and the topological description of how components are interconnected. Each components has a fixed number of ports. Components interact only by being connected at ports.

Each parameter is associated with one port of one component. The behavior of a

component is entirely characterized in terms of constraints that it imposes on values of parameters at its various ports. The lawful behavior of a component is expressed as a set of confluence equations. The behavior of a connection is entirely characterized in terms of constraints that it imposes on the ports that it joins.

2.3.2 Structure to function

A device consists of physically disjoint parts which are connected together. The structure of the device is described in terms of its components and interconnections. Each component has a type, whose generic model is available in the model library. The approach is to infer the behavior of a physical device from a description of its physical structure. The task is to determine the behavior of a device given its structure and access to generic models in the model library. The requirement that models in the library components should be applicable for analysis of different systems places heavy demands for modularity in the component models. This requirement is taken care of through the "No Function In Structure (NFIS)" principle. This states that the laws of the parts of a device may not presume the functioning of the whole. Consider an electrical switch as an example. A model of a switch which states that the current is flowing when the switch is closed violates the NFIS principle because there are many closed switches through which current does not necessarily flow (such as two switches in series). Current flows in the switch only if the switch is closed and there is a potential difference for current to flow.

2.3.3 Deriving behavior

In addition to the confluences describing the components of a physical system, the topology of the system constrains its behavior. Two principles, continuity and compatibility, are used to create additional confluences relating to topology. The continuity principle is applicable for stream-like processes such as electrical current and flows of liquids. It states that the sum of all material flows entering a connection is zero. Since connections are not allowed to accumulate material, this is a sound physical principle.

The compatibility principle is applicable for pressure-like variables. It states that whichever path is taken between two points in the topology, the sum of the pressure drops along different paths must be equal. For electrical systems, this is analogous to Kirchoff's voltage law.

Applying both continuity and compatibility principles to the fullest extent possible produces a redundant set of confluences. This may present difficulties when deriving behavior. Therefore, Envision restricts the number of confluences generated from the topology. It includes only one continuity confluence for every component, and a compatibility confluence for every three conduits [6].

To understand the behavior of a device which is derived from its structures, Envision decomposes a device's behavior into two dimensions, one being interstate behavior and the other being intrastate behavior. The intrastate behavior concerns itself with the behavior within a qualitative state, i.e. the change of values of derivatives while the system remains in a specific state. One or more consistent sets of values of the derivatives are the result from the intrastate analysis. Interstate behavior concerns the possible transitions between states. The consistent sets of values of the derivatives generated from intrastate analysis are used by the interstate analysis to extrapolate from present values to the next qualitative value for each variable, then continues on to reason about which change will occur first. Thus the transition to the next qualitative state is predicted. An example of a pressure regulator that illustrates

the mechanisms of Envision is described in detail in [6].

2.4 Dimensional Analysis

Bhaskar and Nigam [2] introduce an approach for reasoning about physical systems or devices without explicit knowledge of the physical laws that govern the operation of such devices. Their method requires knowledge of the relevant physical variables and their dimensional representation. The dimensional representations of physical variables encode a significant amount of the physical processes, and they can be obtained without explicit knowledge of the underlying laws of physics. A variety of partial derivatives are computed to characterize the behavior of the system. These partials are used to reason qualitatively about the behavior of devices and systems.

To use dimensional analysis as a method for qualitative reasoning about physical systems, Baskar and Nigam developed "regimes". Regimes are a conceptual machinery for reasoning with dimensionless numbers, using elementary notions about partial differentiation. Their method is useful for tackling qualitative reasoning problems, such as the following [2]:

- to resolve, under certain circumstances, some ambiguities inherent in reasoning with a $\{+, 0, -\}$ qualitative calculus;
- to provide a comparative qualitative representation for a physical process;
- to derive the causal structure of a device's behavior, given the inputs and the outputs of a device.

The principle of dimensional homogeneity in all physics can be stated as follows. If

$$y_i = \sum_i a_i x_i \tag{2.3}$$

is a physical law or equation, then $a_i\ x_i$ must have the same dimensions as y_i. If the a_i are dimensionless constants, then each x_i must have the same dimensions as y_i.

Buckingham's theorem [27] tells us that it is possible to extract $n-r$ dimensionless products (Πs) to represent a physical system, where n and r represent the number of variables and the number of dimensions describing the physical system, respectively.

A dimensionless product Π has the following form :

$$\Pi_i = y_i\ x_1^{\alpha_{i1}} \cdots x_r^{\alpha_{ir}} \tag{2.4}$$

where $\{x_1 \cdots x_r\}$ are the repeating variables (variables describing a physical system) , $\{\ y_1 \cdots y_{n-r}\ \}$ are the performance variables and $\{\ \alpha_{ij} | 1 \leq i \leq n-r, 1 \leq j \leq r\}$ are the exponents. A basis is the set of variables x_j that repeat in each Π. A term Π_i is a regime, which refers to a particular physical aspect of the system. An ensemble is a collection of regimes. If a system has n variables and a dimensional matrix of rank r, then the ensemble contains $n-r$ regimes. The dimensional matrix of rank r is defined as a matrix with columns corresponding to the basic dimensions and rows corresponding to the variables. A pivot or contact variable is a variable, x_k, that occurs in both Π_i and Π_j.

The regime Π_i offers us a dimensionally homogeneous equation connecting the variable y_i with the basis variables $x_1, x_2, \cdots, x_r$. From the dimensionless product of equation (2.4), we get

$$y_i = \Pi_i\ x_1^{-\alpha_{i1}} \cdots x_r^{-\alpha_{ir}}$$

where $1 \leq i \leq n-r$. There are three kinds of regimes that are used for reasoning about the behavior of a device or a physical system.

1. Intra-regime partials. They are used for examining how the variables within a regime are related to one another. From the expression obtained from a regime, the change with respect to a basis variable can be obtained by

$$\frac{\partial y_i}{\partial y_j} = -\frac{\alpha_{ij} y_i}{x_j}.$$

2. Inter-regime partials. They are used to relate performance variables y_i and y_j that occur in the regimes Π_i and Π_j respectively. The inter-regime partial models the changes in y_i and y_j in response to a change in contact variable x_p. The notation for an inter-regime partial is

$$[\frac{\partial y_i}{\partial y_j}]^{x_p} = \frac{\frac{\partial y_i}{\partial x_p}}{\frac{\partial y_j}{\partial x_p}}$$

From the regime Π_i, $\frac{\partial y_i}{\partial x_p} = -y_i\frac{\alpha_{ip}}{x_p}$ and from the regime Π_j, $\frac{\partial y_j}{\partial x_p} = -y_j\frac{\alpha_{jp}}{x_p}$ thus the inter-regime partial $[\frac{\partial y_i}{\partial x_p}]^{x_p}$ /$[\frac{\partial y_j}{\partial x_p}]^{x_p} = (\frac{\alpha_{ip}}{\alpha_{jp}})(\frac{y_i}{y_j})$.

3. Inter-ensemble partials.

   Inter-ensemble partials are used to reason about the behavior of a device or system consisting of coupled components or subsystems.

When dealing with a device with several components, the ensemble of each component or subsystem should be obtained. In order to reason about the behavior of the entire device, we need to reason about coupling, which manifests itself in terms of coupling quantities which are used to obtain a coupling regime.

In order to obtain inter-ensemble partials we need contact regimes (a generalization of contact variables). Consider two ensembles A and B, regimes Π_{Ai} and Π_{Bj} belonging to these ensembles and variables y_{Ai} and y_{Bj} that are described by the regimes. The notation for an inter-ensemble partial is

$$[\frac{\partial y_{Ai}}{\partial y_{Bj}}]^{\Pi_C}$$

where Π_C is the contact regime.

When reasoning about the behavior of a system, the objective is to compute the direction of change of a performance variable in response to a change in the basis variable(s). To reason about change in a performance variable y_i as a result of a change in some variables x_j, the following can be used:

- If x_j is in the basis and occurs in Π_i, then use intra-regime partials.

- If x_j is in the basis but not in Π_i, then reason using chains of inter-regime partials.

- If x_j is not in the basis, then use the appropriate inter-regime partial linking Π_i and Π_j.

Examples of analyzing the behavior of physical systems through regime analyses, including a pressure regulator and spring systems, are discussed in detail in [2].

2.5 Hybrid Phenomena Theory (HPT)

Woods [36] presented a Hybrid Phenomena Theory (HPT) which inherited basic concepts like views, phenomena and influences from QPT. HPT defines these concepts in a more precise manner in order to represent physics knowledge with the accuracy needed to develop full parametric models. Parameters have quantitative interpretations which are required by the majority of application areas in science and engineering.

2.5.1 Components of HPT

HPT has three models of components; topological, phenomenological and state space models. Each model is discussed in turn below.

The term topological model is used to refer to a set of objects, and a set of logical statements. The objects provide descriptions of entities such as pipes, valves and control values of physical substances. Logical statements describe properties of these objects and the topological relationships which exist between them. Any given object, as well as any logical statement, may in principle be valid or invalid at a particular point of time. The state of the topological model is thus defined by those objects

and relationships which are valid at that particular point of time. This model is also called the knowledge level of HPT[37].

The term phenomenological model is used to refer to a set of objects describing instances of physical phenomena which may potentially occur in a given system. All objects describing instances of phenomena will not necessarily be active at a given time. The state of the phenomenological model is thus defined by the set of active phenomena instances. This model is also called the symbolic level of HPT[37].

The term state space model is used to derive behavioural predictions in the quantity part of the framework. It is also known as the numeric level of HPT[37]. The fact that an instance of a view or a phenomena exists does not entail that it will influence the analysis of the system. Views describe physical interactions which do not incorporate any dynamic aspects. Phenomena describes physical interactions which do incorporate dynamic aspects. Only the active instances influence the analysis. To be active, all pre- and quantity conditions for an instance must be satisfied. In addition, all objects bound to individuals in the instance have to be active.

To derive the behavioural prediction in the quantity part of a framework, HPT spans a superset of a state space model. This is accomplished by treating the terms of the equations as distinct entities which, at a given point of time, are included in the state space model at that point of time. After having active instances at a given point of time, by combining all influences defined by these instances, HPT will produce a set of equations describing the behaviour of the system.

A HPT model refers to the combined model comprising the topological model, the phenomenological model, the state space model and a set of relationships describing how the state and validity in the respective component models affect the state and validity in the other component models.

HPT generalizes and extends QPT. QPT employs qualitative constraints to describe the relation between variables, while HPT utilizes parametric state space models. Provided that the values of the parameters of the model are known or can be

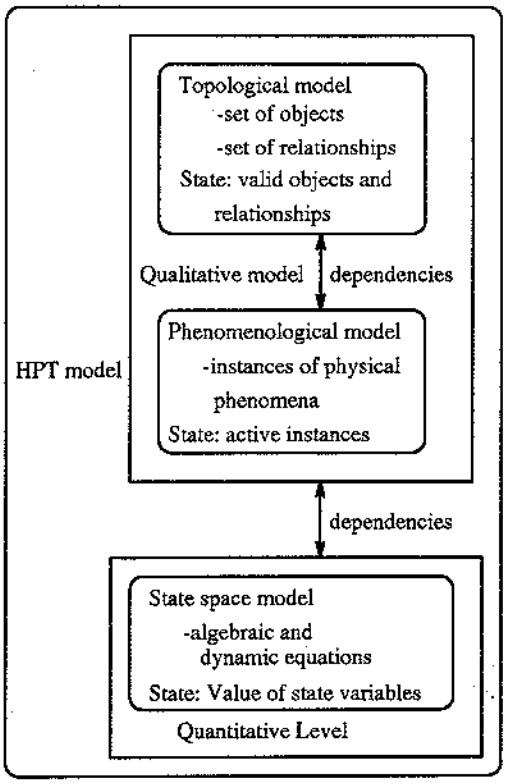

Figure 2.2: Model components in HPT.

estimated, this will allow us to avoid problems like ambiguous solutions which are inherent in the qualitative simulation techniques [36]. Fig. 2.2 shows the model components in HPT. The qualitative model includes both the topological and phenomenological components. In addition, the qualitative model incorporates certain dependencies between topological and phenomenological components which are not attributed to either component. Similarly, in addition to the qualitative and quantitative components, the HPT model incorporates a set of dependencies among these components which are not considered to belong to either component.

The most significant difference between HPT and QPT lies in the fact that HPT employs quantitative values to characterize the properties of the objects, whereas QPT employs qualitative values only. Apart from this, the actual objects used for all practical purposes are identical in both approaches. In addition, the HPT logical

model introduces a type of dependency called subsumption between the entities in the phenomenological model. Subsumption has no counterpart in QPT.

HPT derives parametric state space models, which provides parameters that may have a quantitative interpretation. The concept of an equation is important in HPT, since a large class of problems involving the physical world depend on quantitative knowledge. Influences specify what can cause a quantity to change. A direct influence describes how the value of one quantity influences the derivative of another. A change in the influencing quantity only directly affects the value of the influenced quantity.

2.5.2 Influences in HPT

In HPT, influences express how the value of a given variable is affected by one or a set of other variables. There are two types of influences; dynamic influences and algebraic influences. The former corresponds to the direct influence of QPT and the latter corresponds to indirect influence. Their syntax is

$$\begin{aligned}(dyn-inf \quad & < influenced\ variable > \\ & (< list\ of\ influencing\ variables >) \\ & (< numeric\ function >))\end{aligned}$$

and

$$\begin{aligned}(alg-inf \quad & < influenced\ variable > \\ & (< list\ of\ influencing\ variables >) \\ & (< numeric\ function >))\end{aligned}$$

respectively. An example where a variable x_1 equation is dynamically influenced by two variables x_2 and x_3 is shown in equation (2.5). Here, the amount of influence is computed as a nonlinear function of x_2 and x_3. If this is the only influence affecting

x_1, the derivative of x_1 can be computed from equation (2.6).

$$(dyn - inf\ x_1(x_2x_3)(sqrt(x_2x_3))) \tag{2.5}$$

$$\dot{x_1} = \sqrt{x_2x_3} \tag{2.6}$$

The complete semantic interpretation for dynamic influences is that the derivative of a dynamically influenced variable equals the sum of the numeric functions specified in the dynamic influences affecting the variable. Each dynamic influence is interpreted as the specification of a term in the equation defining the derivative of the dynamically influenced variable.

For algebraic influences mentioned above, the semantic interpretation is that an algebraically influenced variable equals the sum of the numeric functions specified in the algebraic influences affecting that variable. Each dynamic influence is interpreted as the specification of a term in the equation defining the value of the algebraically influenced variable.

To illustrate how influences are combined, consider a closed container which is partially filled, with liquid and gas taking up the remaining volume. The liquid has level l, the gas a pressure p_1. The pressure at the bottom of the container, p_2, is affected by two different influences as shown below.

$$\begin{gathered}(alg - inf\ p_2(p_1)(p_1)) \\ (alg - inf\ p_2(l)(\rho\ g\ l))\end{gathered} \tag{2.7}$$

If no other influence specifies p_2 as influenced variable, this implies that p_2 is given by the expression

$$p_2 = p_1 + \rho\ g\ l$$

where ρ is the liquid density, and g is the earth gravitation.

There are several differences between the algebraic and dynamic influences of HPT and the indirect and direct influences of QPT. The first is that the HPT-influences

are no longer restricted to relate just two quantities. The second is that the HPT influences define complete non-linear functions of any number of variables. The third is that HPT will combine algebraic influences in the same manner as it combines dynamic influences. QPT is modular with respect to dynamic variables; it computes the value of the derivative of all directly influenced variables as the sum of all influences affecting it. QPT is not fully modular with respect to dependent variables. Although several views and processes may specify indirect influences affecting a given variable, QPT fails to specify a general mechanism for deriving the value of an indirectly influenced variable whenever two or more influences are pushing it in different directions. HPT is fully modular in both kinds of variables.

2.5.3 Subsumption

HPT incorporates a mechanism called subsumption, which enables us to avoid conflicts between several instances of the same view and phenomena definitions. Subsumption is defined as follows. For any two instances $INS1$ and $INS2$ of a given view or phenomenon definition with individuals $D1D2\cdots Dm$, $INS1$ subsumes $INS2$ if and only if, for any individual Di $(i = 1\cdots m)$, the object bound to Di of $INS1$ is an instance $INSB$ of a view or phenomenon definition such that the object bound to Di of $INS2$ is also bound to an individual INSB, and the object bound to each of the other individuals in $INS1$ is identical to the object bound to the same individual in $INS2$. Individuals are considered as variables participating in the view or phenomena definitions. Subsumed objects may still be bound to individuals in other instances. For example, consider the physical system shown in Fig. 2.3; i.e. a pan containing an amount of water placed on a hot plate. We want to model the effects of heat flow from the hot plate to the water. There are six possible instances *heat-flow1 (pan hotplate), heat-flow2 (hotplate pan), heat-flow3 (water pan), heat-flow4 (pan water), heat-flow5 (container-with-liquid (pan water) hotplate)*, and *heat-flow6 (hotplate*

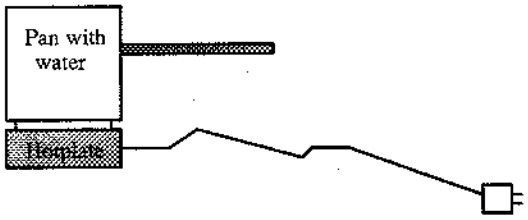

Figure 2.3: A pan with water on hotplate.

container-with-liquid (pan water)). These instances will not all be active at the same time. The quantity conditions affected by temperature will allow instances *heat-flow2 (hotplate pan) and heat-flow6 (hotplate container-with-liquid (pan water))* to be active at the same time. The subsumption mechanism works from the principle that the most specific alternative prevails. In this example the instance view definition of *container-with-liquid (pan water))* is more specific than the *pan* as such, therefore the *heat-flow6 (hotplate container-with-liquid (pan water))* subsumes the *heat-flow2 hotplate pan)*.

Examples of deriving HPT models for physical systems, such as the pan with water on a hot plate and a three-piece-insulator, are discussed in detail in [38].

2.6 Comparison

From the discussions presented in the previous sections it can be summarized that:

1. Both QPT and HPT provide a framework, or vocabulary, to formalize knowledge of physical interactions. This is the idea of views and phenomena definitions comprising a prescription for relations and influences between individual quantities which will hold for instances of the definition. In addition, QPT includes a set of view and process definitions which may be instantiated. Envision also provides such a framework; this is the idea of generic component models, each consisting of a number of confluences. Each confluence describes a component in

a specified qualitative state. In addition, Envision includes a library of generic component models.

2. Both QPT and HPT have a mechanism which produces a set of constraints describing the system in the current state. The mechanism consists of two steps. First, the view and process definitions for each set of objects satisfying the individual conditions for each definition are instantiated. Next, a process structure consisting of instances whose pre- and quantity conditions are satisfied in the current situation is established, and the influences from these instances are extracted. Envision also has a similar mechanism which consists of two steps. First, the compatibility and continuity constraints are generated. Next, the confluences which describe each component in the system's current qualitative state are selected.

3. QSIM provides neither a framework to formalize knowledge on physical interactions nor a mechanism such as mentioned in 2. QSIM does, however, formulate the problem in terms of constraints to be used. QSIM does not provide assistance in modeling the system except for those instances when a model in terms of Ordinary Differential Equations (ODEs) exists, in which case a procedure for converting ODEs to constraints may be applied. Envision provides a library of components. The user needs only to specify what kind of components the system is built from, and how the components are interconnected for Envision to select the right confluences modeling system. QPT and HPT are similar in that the user only specifies which objects take part and the relations existing between these objects.

4. QPT, HPT and Envision use the set of generated constraints to derive possible successor states. A new set of constraints is produced whenever a new qualitative state is reached. This approach is iterated until quiescence occurs or until the successor state equals a previously identified state. Only HPT produces a

mathematical model of each state.

5. Envision and QSIM describe behavior in terms of qualitative state while QPT and HPT describe behavior in terms of process structures. QPT and HPT make use of notions of phenomena explicitly and relate behavior to the occurrence of physical phenomena. Neither QSIM nor Envision embodies a counterpart to the notion of phenomena. Therefore, QPT and HPT provide us with a stronger conceptual framework for formalizing our knowledge about physical systems. Through the introduction of pre- and quantity conditions, QPT allows us to formalize conditions for when these phenomena occur. The QSIM approach only addresses the aspects related to simulation, while Envision, QPT and HPT address the additional issue of deriving a model, expressed as constraints, from a topological description of a process.

6. HPT generalizes and extends QPT. QPT employs qualitative constraints to describe relations between variables. HPT utilizes a parametric state space model. Among the approaches reviewed, only HPT has a representation which allows us to modify mathematical models in accordance with the actual situation.

7. None of those approaches can represent nuclear physical processes, since none of them has a mechanism capable of reasoning about probabilistic events such as occur in nuclear physical processes.

A comparison of physics reasoning systems reviewed here is shown in Table 2.1.

Table 2.1: Comparison of physics reasoning systems.

Property	QSIM	Envision	Dimensional	QPT	HPT
input	constraint	component and topology	variables and dimensions	objects and relations	objects and relations
model is derived from	differential equations	qualitative equations of its components and interconnections	quantities' dimensions	topological description of process	topological description of process
constraints	M^+, M^- Add, Mult, Deriv	confluences	regimes	direct and indirect influences	algebraic and dynamic influences
mechanism producing constraints	derived from differential equations	continuity compatibility and NFIS	derived from physical quantities' dimensions	-instantiate view and process definitions -establish Process Structure -extract influences	-instantiate view and process definitions -establish active processes -extract influences
quantity value	dec,0,inc	-,0,+	dec, 0, inv	-1,0, 1	R
represent state at discrete times	y	y	n	y	y
represent state variables	n	n	n	y	y
represent a feedback system	y	y	y	y	y
quantitative result	n	n	n	n	y
modularity	n	y	n	y	y
library	n	y	n	y	y
computer program	QSim	Envision	n	GIZMO	HPT
focus	mechanics	thermo-dynamics	mechanics	thermo-dynamics	thermo-dynamics

Chapter 3

Nuclear Processes

Nuclear processes occur when a nucleus is bombarded by an energetic particle [21]. The nucleus may undergo a transformation to a new nuclide accompanied by the ejection of a particle. The particle released may be different from the incident one. A certain amount of energy is released from both the product nuclide(s) and the product particle(s). A nuclear process can be expressed as

$$a + T \longrightarrow P + b + E_P + E_b \tag{3.1}$$

where T and P are the target and the product nuclide(s), respectively; a and b are the incident and product particle(s), respectively. E_P and E_b are the energy released from the product nuclide(s) and product particle(s), respectively. A nuclide is characterized by the number of protons (atomic number) Z, and the number of neutrons N in the nucleus. The total number of protons and neutrons in a nucleus is equal to the atomic mass number A of the nuclide.

A very wide range of nuclear reactions have been observed experimentally. Of these, the reactions of most interest to the study of nuclear reactors are the ones that involve neutrons. If a neutron interacts with a nuclide, a compound nucleus is formed. The compound nucleus then undergoes several possible processes, such

as fission, radioactive capture, elastic scattering, and inelastic scattering. The most important nuclides for nuclear reactions are summarized in [21], and are shown in Fig. 3.1. In this table, isotopes are in rows with horizontal arrows representing radioactive capture. The downward arrows represent β^- decays; the β^+ decay is not shown. The diagonal dashed arrows represent thermal-neutron fission. Decay half-lives and thermal cross sections are from the Chart of the Nuclides [31]. There are three types of nuclear processes shown in the figure: fission, radioactive capture, and beta negative decay. It can be seen from Fig. 3.1, for example, that $Th233$ can interact with neutrons through fission and radioactive capture processes. It decays through beta negative decay processes. These interactions are indicated by its properties such as fission cross section, radioactive capture cross section and beta decay mode.

For fission, T is a heavy target nucleus and the products of a fission reaction are two fission products F_1 and F_2. The incident particle is a neutron $({}^1_0n)$. The ejected particles b are neutron and gamma $({}^0_0\gamma)$ particles. The mass and atomic number of the fission products F_1 and F_2 do not have deterministic values. The gamma energy and fission neutron energy follow energy distributions called the prompt-gamma-spectrum and the prompt-neutron-spectrum [25]. For a typical target nuclide ${}^{235}_{92}U$, the mass number of fission product F_1 ranges from 73 to 114, and from 115 to 162 for fission product F_2. An example fission process is given as follows:

$$ {}^{235}_{92}U + {}^1_0n \longrightarrow {}^{90}_{36}Kr + {}^{144}_{56}Ba + 2{}^1_0n + 168 \text{ MeV } + 5 \text{ MeV } + 7 \text{ MeV } . $$

A radioactive capture process occurs when a particle combines with a nucleus to produce a new nucleus. The excess energy is emitted in the form of γ rays. An example of this is

$$ {}^{235}_{92}U + {}^1_0n \longrightarrow {}^{236}_{92}U + {}^0_0\gamma + E_\gamma $$

where $E\gamma$ is a non-deterministic number averaging 1.98 MeV.

Beta negative decay occurs for nuclei in which there is an excess of neutrons. A neutron inside a nucleus may be converted to a proton, with emission of an electron

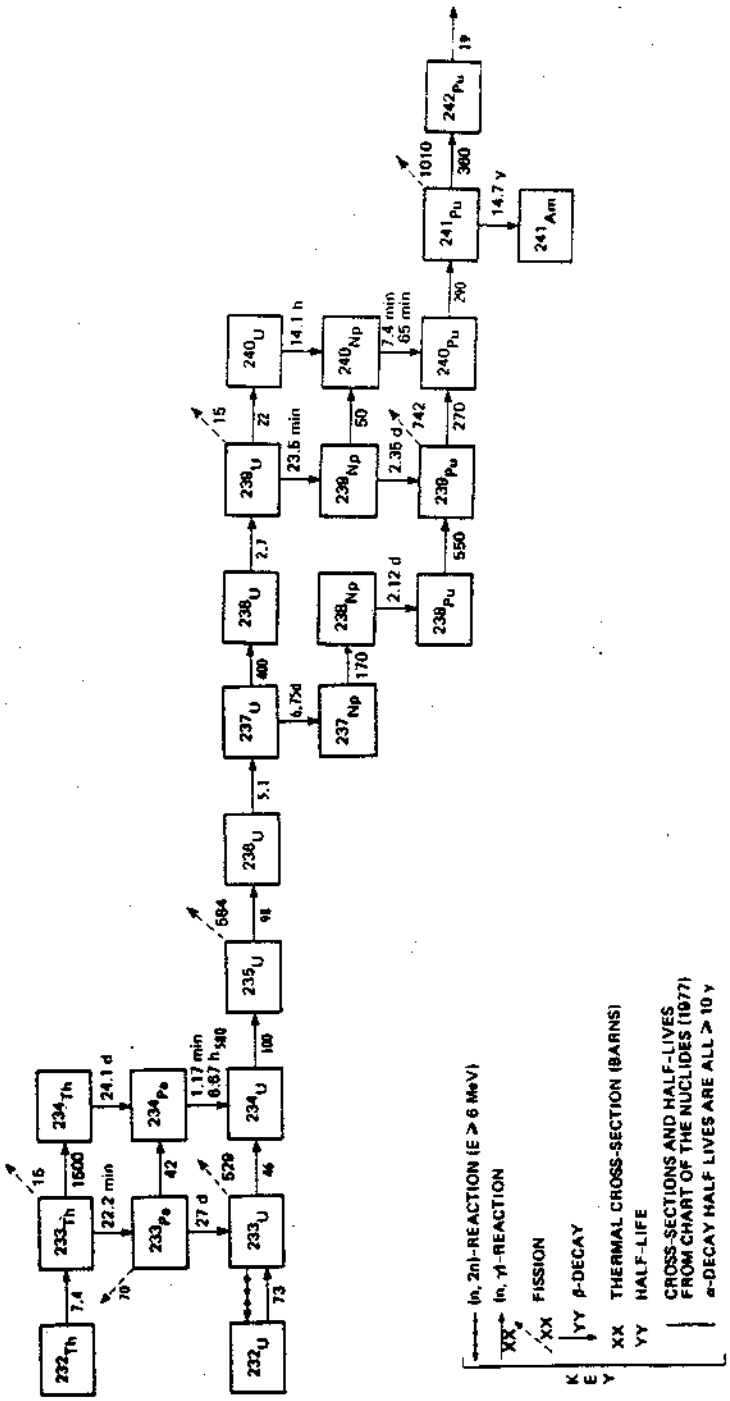

Figure 3.1: Heavy elements of interest for nuclear reactors from [21].

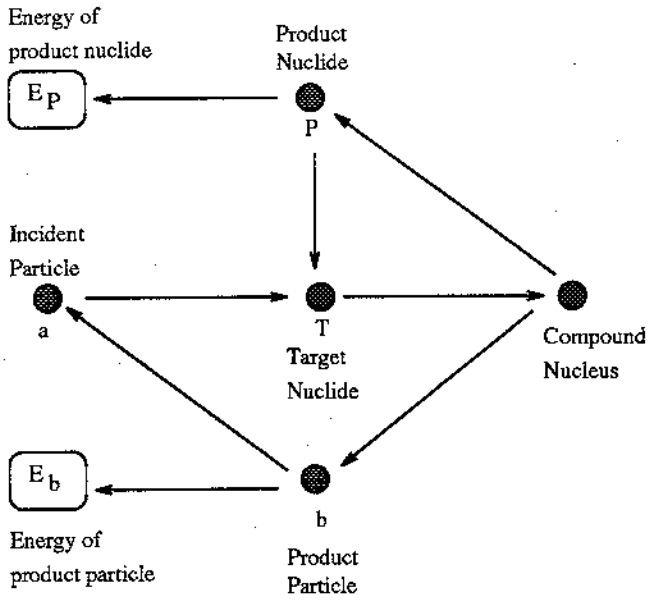

Figure 3.2: A simplified diagram for nuclear physics processes.

($^{0}_{-1}e = \beta^{-}$). The nuclear charge is consequently increased by one, with the mass number being unchanged. For example, a beta negative reaction

$$^{135}_{55}Cs \longrightarrow ^{135}_{56}Ba + \beta^{-}$$

results in a new nuclide $^{135}_{56}Ba$.

Beta positive decay occurs for nuclei in which there is a deficiency of neutrons. A proton inside a nucleus may be considered to be converted into a neutron, with emission of a positron ($^{0}_{+1}e = \beta^{+}$). The resultant atomic number is decreased by one, and the mass number remains unchanged. For example, a reaction

$$^{141}_{60}Nd \longrightarrow ^{141}_{59}Pr + \beta^{+}$$

produces $^{141}_{59}Pr$ as a new nuclide.

The nuclear processes described above are shown schematically in Fig. 3.2. In this diagram

$$T \in \{\text{possible nuclides}\}$$

$$P \in \{\text{possible nuclides}\}$$

$$a \in \{\ {}^{1}_{0}n, {}^{2}_{1}d, {}^{1}_{1}p({}^{1}_{1}H)\ \}$$

$$b \in \{\ {}^{1}_{0}n, {}^{0}_{0}\gamma, \beta^{+}, \beta^{-}, \alpha({}^{4}_{2}He), {}^{1}_{1}p\ \}$$

The arrow pointing from P to T shows that product nuclides may become target nuclides within a cycle period $[t_i, t_{i+1}]$. The cycle period is discussed further in the next section. Similarly, the arrow pointing from b to a shows that the product particles can be incident particles in the interval $[t_i, t_{i+1}]$. Both T and P are nuclides in the chart of nuclides data [18].

3.1 Behaviour of Nuclear Physics Processes

Unlike a typical physical system, a nuclear physics process q is not only characterized by the number of real-valued parameters, which vary continuously over time, but is also characterized by the type of incident particle, and the type of target nuclide that is involved in the process. In addition, a residual nuclide and particle along with their released energy also characterize the nuclear physics process. There are about 3000 different types of nuclides that might be involved in the nuclear process.

The quantitative behaviour of q over an interval of time $[t_i, t_{i+1}]$ is the sequence of individual state space and probabilistic space equations of q, which are generated by j different types of target nuclides T. The notation SP describes a collection of state space and probabilistic space equations, as follows:

$$SP(q, [t_i, t_{i+1}]) = SP(q, T_1, [t_i, t_{i+1}]), SP(q, T_2, [t_i, t_{i+1}]), \cdots, SP(q, T_j, [t_i, t_{i+1}])$$

A nuclear physics system is a set $Q = q_1, q_2, \cdots q_m$ of nuclear physics processes. The quantitative behaviour of nuclear physics processes Q during interval $[t_i, t_{i+1}]$ is described by the sequence of j sets of the combined sets of equations, as follows:

$$SP(Q, [t_i, t_{i+1}]) = SP(q_1 \cdots q_m, T_1, [t_i, t_{i+1}]), SP(q_1 \cdots q_m, T_2, [t_i, t_{i+1}]),$$

$$\cdots, SP(q_1 \cdots q_m, T_j, [t_i, t_{i+1}]),$$

where j refers to the number of types of nuclides available in the system. The combined state space and probabilistic space equations $SP(q_1 \cdots q_m, T_r, [t_i, t_{i+1}])$ is generated from individual state space and probabilistic space equations, when an activating nuclide of type T_r activates the processes Q, i.e.

$$SP(q_1 \cdots q_m, T_r, [t_i, t_{i+1}]) = SP(q_1, T_r[t_i, t_{i+1}]), SP(q_2, T_r, [t_i, t_{i+1}]),$$

$$\cdots, SP(q_m, T_r, [t_i, t_{i+1}])$$

The quantitative behaviour of Q over time interval $[t_0, t_n]$ is the sequence of state and probabilistic space equations SP of Q which come from an active target nuclide T_r, i.e.

$$SP(Q, T_r, [t_0, t_1]), SP(Q, T_r, [t_1, t_2]), \cdots, SP(Q, T_r, [t_{n-1}, t_n])$$

where $SP(Q, T_r, [t_i, t_{i+1}])$ refers to a set of probabilistic and deterministic equations related to the nuclear physics processes Q. The equations show how to obtain the type and amount of product nuclide P and product particles b, and the amount of energy E_P and E_b released from product nuclides and from product particles, respectively. In the example reported in this thesis, the neutron life time is adopted as the time interval $[t_i, t_{i+1}]$ which is also referred to as a cycle time.

3.2 Neutron Cycle

This section discusses a neutron cycle for thermal reactors. In a thermal reactor, fast neutrons are born of fissions. The possibility of a chain reaction was recognized as soon as it was known that neutrons are released in a fission. Neutrons emitted by fissioning nuclei induce fissions on other fissile or fissionable nuclei. Neutrons generated by these induced fissions, cause fissions in other fissile or fissionable nuclei, and so on. Such a

chain reaction can be described quantitatively in terms of a multiplication factor, k [25].

The multiplication factor is defined as the ratio of the number of neutrons in the current generation, n_g, to the number of neutrons in the previous generation, n_{g-1}, i.e.

$$k = n_g \, (n_{g-1})^{-1}.$$

The value of k indicates the criticality of a reactor. If $k > 1$ the number of fissions increases from generation to generation. The reactor is said to be *supercritical* and the power it generates rises exponentially. On the other hand, if $k < 1$ the reactor is said to be subcritical and the power it produces decreases with time. If $k = 1$ the reactor produces constant energy and the reactor is said to be in its critical stage.

The multiplication factor k can also be expressed in terms of the four-factor formula [25] as follows

$$k = \eta \, \epsilon \, p \, f \, P_{NLf} \, P_{NLth}, \tag{3.2}$$

where η is defined as the average neutrons emitted per neutron absorbed in fuel or $\eta = (\sigma_f/\sigma_a)\,\nu$, where ν is the average number of neutrons released per fission. This set of reactor parameters are *fast fission factor* ϵ, *resonance escape probability* p, *thermal utilization factor* f, and nonleakage probability which is subdivided into *fast nonleakage probability* P_{NLf}, and *thermal nonleakage probability* P_{NLth}. These factors are defined in [25].

In the NPT model developed here, which is discussed in Chapter 4, a fission chain is initialized by a number of neutrons $n_{g-1} = X$ in the reactor core. The number of neutrons increases slightly after some of the neutrons are involved in fast fissions resulting in a total number of fast neutrons of $n_{g-1}\,\epsilon$(see Fig. 3.3). Each fast neutron has a probability of remaining in the reactor core of P_{NLf} leaving $n_{g-1}\,\epsilon\,P_{NLf}$ neutrons remaining in the core after allowing for fast leakage. These fast neutrons scatter and slow down. A $(1-p)$ fraction of the neutrons are captured in a resonance capture process. The remaining $n_{g-1}\,\epsilon\,P_{NLf}\,p$ escape from resonance capture

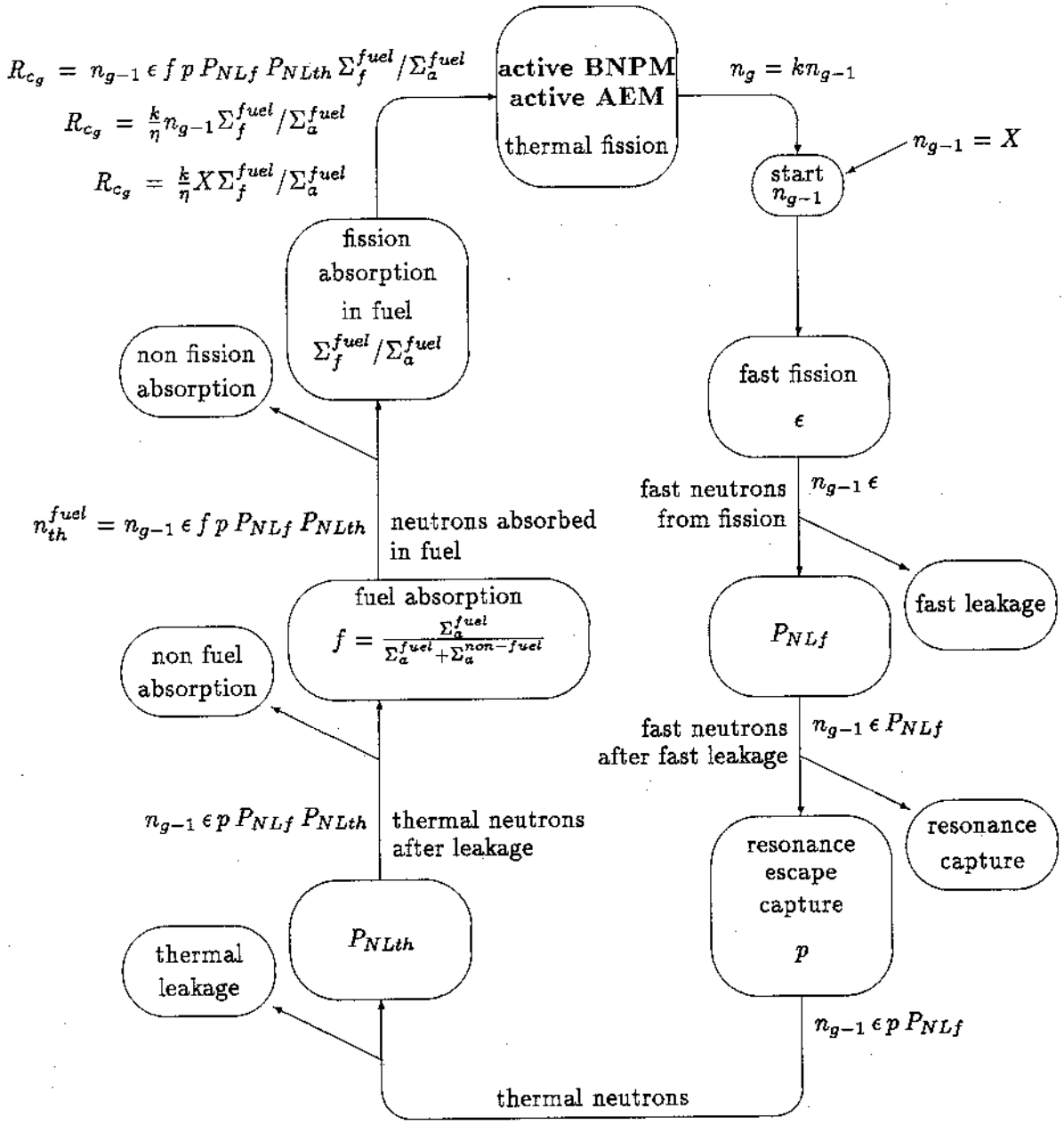

Figure 3.3: Neutron cycle in NPT.

during slowing down and are thermalized. A fraction of the thermalized neutrons, $n_{g-1}\,\epsilon\,P_{NLth}\,p\,P_{NLf}$, remain in the core after diffusion at thermal energy. A fraction of these neutrons is absorbed in non-fuel material leaving $n_{g-1}\,\epsilon\,P_{NLth}\,p\,P_{NLf}\,f$ neutrons absorbed in fuel. The fraction of thermal neutrons absorbed in fuel is denoted as n_{th}^{fuel} which is equal to

$$n_{th}^{fuel} = n_{g-1}\,\epsilon\,P_{NLf}\,p\,P_{NLth}\,f. \tag{3.3}$$

Substituting equation (3.2) into (3.3) results in

$$n_{th}^{fuel} = k\,\eta^{-1}\,n_{g-1}.$$

Some of these neutrons are involved in fission processes while others participate in radioactive capture processes. Since a process is instantiated by nuclides of a certain type, an expression for distributing n_{th}^{fuel} is needed. This subject will be discussed later. For now, let us assume that there is only one type of nuclide and that this nuclide can participate in both fission and radioactive capture. The fraction of thermal neutrons participating in a fission process, F_f, is computed as

$$F_f = \Sigma_f^{fuel}\,(\Sigma_a^{fuel})^{-1}$$

Σ_f^{fuel} and Σ_a^{fuel} are the macroscopic fission and absorption cross sections, respectively. They are defined in [13]. Therefore, the number of fissions at the current cycle is R_{c_g} $= n_{th}^{fuel}\;F_f$ which is equal to

$$R_{c_g} = k\,\eta^{-1}\,n_{g-1}\,\Sigma_f^{fuel}\,(\Sigma_a^{fuel})^{-1}, \tag{3.4}$$

as shown in Fig. 3.3.

Since $n_{g-1} = X$ represents the number of fission neutrons in the previous generation, equation (3.4) becomes

$$R_{c_g} = k\,\eta^{-1}\,X\,\Sigma_f^{fuel}\,(\Sigma_a^{fuel})^{-1}. \tag{3.5}$$

The majority of the neutrons available in the next cycle come from fission processes.

Each cycle of Fig. 3.3 represents a generation of prompt neutrons, from their emission until their absorption in fuel. The average time that the neutrons spend from their emission until their absorption is denoted as the *prompt neutron lifetime* [25]. The terms BNPM and AEM shown in Fig. 3.3 are discussed in Chapter 4.

The amount of fuel consumed, and the energy released in a reactor core are computed every cycle, as well as the composition of fission products. Furthermore, the number of nuclides of each fission product is updated every cycle time t_c due to decay chains. This will be explained separately.

Chapter 4

Nuclear Process Theory (NPT)

Nuclear Process Theory (NPT), a formal representation system for nuclear physics processes, was developed. Knowledge of the nuclear physics process is expressed in terms of a basic nuclear physics model and aggregate effects model. The basic nuclear physics model has the capability of expressing knowledge of nuclear physics processes for different types of nuclear reactions. The products of reactions are expressed as deterministic or probabilistic values. The aggregate effects model coordinates the basic nuclear physics processes expressed in the basic nuclear physics model. NPT allows the explicit representation of knowledge of nuclear physics interactions, and formalizes how the interactions affect the structure and parameters of the model. The effects on the model are encoded in a symbolic structure. NPT has three different levels, as shown in Fig. 4.1. Each of them is described separately.

4.1 Knowledge Level

The knowledge level incorporates general descriptions of reaction components, materials and nuclear interactions. Two types of knowledge are embodied in this level.

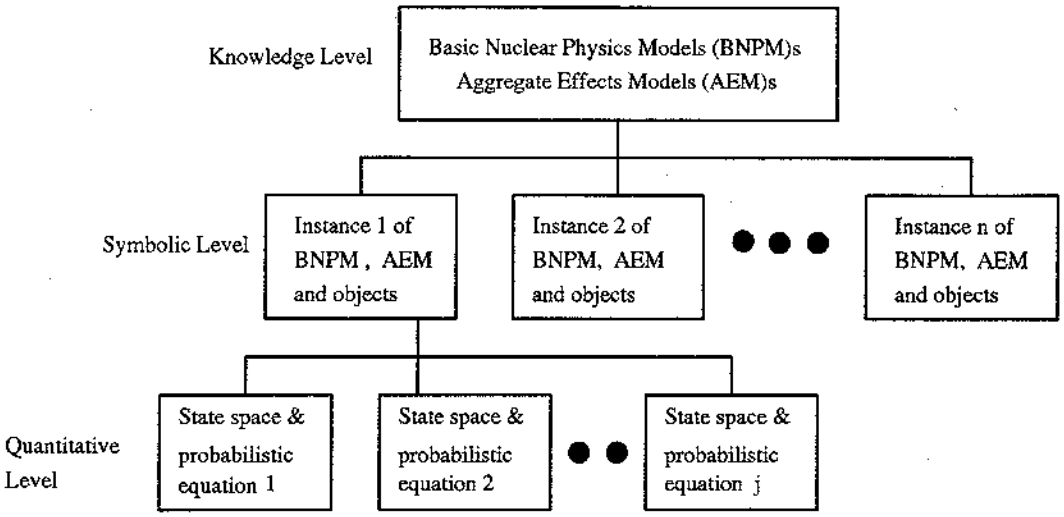

Figure 4.1: Model hierarchy in NPT.

Firstly, there is a description of the properties of various types of material substances. This knowledge is in the representation of the nuclide data [18]. Secondly, there is knowledge about nuclear physics interactions, which is expressed in terms of definitions in the basic nuclear physics model and the aggregate effects model. At this stage, a nuclear physics process is purely abstract, and not related to any objects. The knowledge expressed in the basic nuclear physics model describes a type of nuclear process which is applicable to many types of nuclides.

4.1.1 Basic Nuclear Physics Model (BNPM)

The basic nuclear physics model consists of descriptions of the underlying nuclear process being modeled. It contains reagents, conditions for when a process occurs, product specifications of the process, and it may contain intermediate states in which a process participates. It is characterized by four fields: reactants, activity_conditions, intermediate_state, and products. Examples of basic nuclear physics models for fission, radioactive capture, beta negative and beta positive decay processes, are shown in Figures 4.2, 4.3, 4.4, and 4.5, respectively.

```
basic_nuclear_physics_model fi is fission
{
    reactants {
        obj1 is NUCLIDE
        obj2 is PARTICLE }
    activity_conditions { obj1.sigma_f > 1 & obj2.id = NEUTRON }
    intermediate_state { neutron_absorbtion c_n (obj1) }
    products {
        def_obj { Neutron PARTICLE
            Neutron.id is DETERMINISTIC (value: 32)
            Neutron.amount is PROBABILISTIC
                (value: decomposition(pdf_neutron, Poisson(2.43,2.43,0,8)))
            Neutron.energy is PROBABILISTIC
                (value: look_up_table(pdf_E_neutron, neutron_sp(0,8))) }
        def_obj { F1 NUCLIDE
            F1.Z is PROBABILISTIC
                (value: decomposition(pdf_dZ, Gaussian(0,1,-4,4)))
            F1.A is PROBABILISTIC
                (value: look_up_table(pdf_A1, mass_yieldA1(73,117)))
            F1.energy is PROBABILISTIC
                (value: look_up_table(pdf_E_F1, energy_distrF1(40,84))) }
        def_obj { F2 NUCLIDE
            F2.Z is PROBABILISTIC
                (value: decomposition(pdf_dZ, Gaussian(0,1,-4,4)))
            F2.A is PROBABILISTIC
                (value: look_up_table(pdf_A2, mass_yieldA2(118,162)))
            F2.energy is PROBABILISTIC
                (value: look_up_table(pdf_E_F2, energy_distrF2(85,129))) }
    }
}
```

Figure 4.2: Basic nuclear physics model of fission.

```
basic_nuclear_physics_model rc is radioactive_capture
{
   reactants {
      obj1 is NUCLIDE
      obj2 is PARTICLE }
   activity_conditions { obj1.sigma_g > 1 & obj2.id = NEUTRON }
   products {
      def_obj { P NUCLIDE
         P.Z is DETERMINISTIC (value: obj1.Z)
         P.N is DETERMINISTIC (value: obj1.N+1) }
      def_obj { Gamma PARTICLE
         Gamma.id is DETERMINISTIC (value: 8) }
         Gamma.energy is PROBABILISTIC
         (value: look_up_table(pdf_E_gamma, gamma_sp(0,8))) }
   }
}
```

Figure 4.3: Basic nuclear physics model of radioactive capture.

```
basic_nuclear_physics_model bpd is beta_positive_decay
{
   reactants {
      obj1 is NUCLIDE }
   activity_conditions { obj1.dmode = BETA_POSITIVE }
   products {
      def_obj { Beta_Pos PARTICLE
         Beta_Pos.id is DETERMINISTIC (value: 2) }
      def_obj { P NUCLIDE
         P.Z is DETERMINISTIC (value: obj1.Z-1)
         P.N is DETERMINISTIC (value: obj1.N+1) }
   }
}
```

Figure 4.4: Basic nuclear physics model of beta positive decay.

```
basic_nuclear_physics_model bnd is beta_negative_decay
{
   reactants {
      obj1 is NUCLIDE }
   activity_conditions { obj1.dmode = BETA_NEGATIVE }
   products {
      def_obj { Beta_Neg PARTICLE
         Beta_Neg.id is DETERMINISTIC (value: 4) }
      def_obj { P NUCLIDE
         P.Z is DETERMINISTIC (value: obj1.Z+1)
         P.N is DETERMINISTIC (value: obj1.N-1) }
   }
}
```

Figure 4.5: Basic nuclear physics model of beta negative decay.

reactants - describes what reagents are being used in the reaction being modeled, and the relations between reagents which are necessary for the reaction to occur. In our example, the reagents are represented symbolically by *obj*1 and *obj*2. They are in classes of NUCLIDE and PARTICLE respectively.

activity_conditions - describes conditions which are necessary for a process to be active. It specifies either absolute limits on one variable, or limits relative to two variables. In our example, the radioactive capture process can only be activated by nuclides whose radioactive capture cross section is greater than 1 barn.

intermediate_state - describes actions which occur before the process generates a set of products. For example, the intermediate_state of a fission process is a state where the projectile particle absorbs the target nucleus, forming a compound nucleus.

products - describes a set of products generated as a consequence of an active process. The products are characterized by their class name, along with the descriptions of the quantities which characterize them (e.g. atomic number and

atomic mass number). The values of the quantities may be probabilistic or deterministic depending on the type of reaction itself. The product nuclides P are specified by their atomic numbers $P.Z$ and their neutron numbers $P.N$ (e.g. see Fig. 4.3) in the basic nuclear physics model. Their type and their amount are determined by specifying **cum_infl** in the aggregate effects model (see Fig. 4.8) which is described later.

The products field in the basic nuclear physics model defines the quantity of products generated in a reaction. If the basic nuclear physics model of a specific reaction is active, it generates a set of products symbolically, and their amounts are either PROBABILISTIC or DETERMINISTIC values. The PROBABILISTIC values are represented in a vector, while the deterministic values are represented by real values. The element of the vector represents either the probability of a parameter being in a certain interval or the probability of a parameter having a certain value.

The probability of x being in interval (x_1, x_2) is defined by

$$P(x_1 < x < x_2) = \int_{x_1}^{x_2} f(x)\, dx \tag{4.1}$$

where f(x) represents a probability density function (pdf) [30]. The probability of x having a certain value is mapped by a probability measurement value which is defined by [30]

$$P(X = x) = f(x) \tag{4.2}$$

In the case that the value of a quantity is PROBABILISTIC, its distribution is specified by a specific distribution curve and its range is known. The vector representing the curve must be defined, and the vector values must represent the y axis values of the curve. In our example, when $obj1$ represents $U235$, and $obj2$ represents a neutron, the nuclear reaction for fission resulting in ^{235}U splits into two fragments as follows

$$^{235}_{92}U + {}^{1}_{0}n \rightarrow {}^{A_1}_{Z_1}F_1 + {}^{A_2}_{Z_2}F_2 + \nu\, {}^{1}_{0}n + Energy \tag{4.3}$$

where F_1 and F_2 indicate the products of fission fragments, A_1 and A_2 represent mass numbers of F_1 and F_2, respectively, and Z_1 and Z_2 represent atomic numbers of F_1 and F_2, respectively.

The target nuclide and the incident particle, ${}^{235}_{92}U$ and ${}^{1}_{0}n$, are represented as the elements of **reactants** of Fig. 4.2. The products of fission fragments F_1 and F_2 are represented as the elements of **products** of Fig. 4.2. These are expressed as **def_obj** { F1 NUCLIDE ...} for fission fragment F_1 and **def_obj** { F2 NUCLIDE ...} for fission fragment F_2.

The splitting of a fissioning nucleus does not always produce equal mass numbers. Mass numbers of fission fragments for ${}^{235}U$ range from 72 to 163, with the most probable mass numbers at ranges roughly 90-100 and 135-140 [21]. Symmetric fission, a fission producing equal mass numbers, is relatively uncommon occurring in only about one in 20,000 events [26].

The mass numbers of fission fragments F_1 are probabilistic; they are expressed as F1.A is PROBABILISTIC (value: look_up_table(pdf_A1, mass_yieldA1(73,117))) in Fig. 4.2. Fission fragments F_2 are expressed in a similar way. Since NPT is used for reasoning about nuclear physics processes, the probabilities per unit mass of different fissionable nuclei must be provided. This is a vector with indices representing the range of values of the mass distribution. The values of vector elements are the probability of a fragment having a certain mass number. For example, if the target of fission is ${}^{235}U$, the probabilities of having a fission fragment with a mass number 73, 74 and 75 are 0.0001, 0.0003, and 0.0010, respectively. For each target, there is a pair of mass numbers, and the probability value of producing a fission fragment with that mass number. These pairs are provided as a look-up table called *mass_yield*. Further details are provided in Section 6.1.1.

The product particle of fission, neutron, is represented as one of the elements of **products** in Fig. 4.2. It is expressed as **def_obj** { Neutron PARTICLE ...}. A

neutron is uniquely identified by an id value, and expressed as Neutron.id is DETERMINISTIC (value: 32). Other particles (e.g. proton, photon, gamma, beta-positive, beta-negative) have different id values.

The number of neutrons produced is probabilistic between 0 and 8. It is controlled by the Poisson distribution, which is represented by pdf_neutron. This number of fission neutrons is expressed as Neutron.amount is PROBABILISTIC (value: decomposition(pdf_neutron, Poisson(2.43,2.43,0,8))). The energy of fission products is also probabilistic depending upon the energy distributions. The energy distributions are represented by pdf_E_F1 and pdf_E_F2 for energy of fission fragments F_1 and F_2, respectively. The energy of fission products F_1 is expressed in Fig. 4.2 as F1.energy is PROBABILISTIC (value: look_up_table(pdf_E_F1, energy_distrF1(40,84))) .

4.1.2 Aggregate Effects Model (AEM)

An aggregate effects model is used to aggregate nuclear physics processes expressed in the basic nuclear physics model as a consequence of an active process. The aggregate effects model describes nuclear process interactions in terms of influences which affect parameters of the system over a time period (e.g once per neutron life-time). It is characterized by the fields material, equipment, preconditions, activity_conditions, parametric_reactions, relations, and products for the main process. Examples of aggregate effects models for fission, radioactive capture, beta negative and beta positive decay processes, are shown in Figures 4.6, 4.7, 4.8, 4.9 and 4.10, respectively.

material - describes material being used in a reaction being modeled; it may be the same as one of the reactants in the basic nuclear physics model. In our example, the material is denoted symbolically by **obj1**.

equipment - specifies a place where a reaction is to be considered, along with its components and parameters. In a fission reaction, the equipment could be a reactor with all its parameters.

preconditions describes conditions which may depend on external actions. Turning a power switch on or off is an example. No reaction occurs unless the power is turned on, but once on, the power is toggled depending on the state of the system.

activity_conditions - describes conditions which are necessary for processes to be active. Whenever the conditions are violated, the model is no longer an adequate description of the process. It specifies either absolute limits on one variable, or limits relative to two variables. In our example, the radioactive capture process stops if no amount of obj1 is available in the system.

parametric_reactions - describes a set of parameters which are necessary for the system being modeled. A statement in the parametric field defines a new constant, parameter or vector and associates it with each instance of the definition. Fig. 4.6 shows two definitions of constants, NA and ei, where NA and ei are Avogadro's number and the interval of gamma energy, respectively.

relations - specifies relationships to be associated with each instance of the definitions. The relationships are expressed as influences, such as algebraic, aggregate, cumulative, distribution, and decay influences.

products - specifies the creation of objects (e.g. nuclides or particles) which come into being as a consequence of an active aggregate effects model. An active aggregate effects model causes a set of new objects to be created. These are objects which are stated in the products field of the basic nuclear physics model.

The products field in the basic nuclear physics model represents a product of a process under consideration when a nuclide interacts with a neutron. A statement in the product field in the aggregate effects model aggregates products of a process under consideration when a number of nuclides interact with a number of neutrons.

```
aggregate_effects_model fission(fi)
{
  material { obj1 }
  equipment {
    PHWNUR is REACTOR
    HW is MODERATOR
    Zra is CLADDING }
  preconditions { PHWNUR.power is ON }
  activity_conditions { obj1.amount > 0 & obj2.amount > 0 }
  parametric_reactions {
    def_const NA (value: 6.02217e23)
    def_const ei (value: 1)
    def_const nu (value: 2.43)
    def_vector pdf_E_gamma (value: look_up_table(gamma_sp(0,8)))
  relations {
    alg_infl { (Sigma_a_fuel)(obj1.amount,obj1.sigma_f)(obj1.amount*obj1.sigma_f) }
    alg_infl { (Sigma_f_fuel)(obj1.amount,obj1.sigma_f)(obj1.amount*obj1.sigma_f) }
    alg_infl { (f(obj1))
      (obj1.amount,obj1.sigma_f,obj1.sigma_g,Sigma_a_fuel,Zra.sigma_a,HW.sigma_a)
      (obj1.amount*(obj1.sigma_f+obj1.sigma_g)/
      (Sigma_a_fuel+Zra.sigma_a+HW.sigma_a)) }
    alg_infl { (eta)(nu,obj1.amount,obj1.sigma_f,Sigma_a_fuel)
      (nu*obj1.amount*obj1.sigma_f/Sigma_a_fuel) }
    alg_infl { (k)(f(obj1),eta,
      PHWNUR.epsilon,PHWNUR.p,PHWNUR.PNLf,PHWNUR.PNLth)
      (f(obj1)*eta*PHWNUR.epsilon*PHWNUR.p*PHWNUR.PNLf*PHWNUR.PNLth)}
    alg_infl { (Ff(obj1))(obj1.amount,obj1.sigma_f,Sigma_a_fuel)
      (obj1.amount*obj1.sigma_f/Sigma_a_fuel) }
    alg_infl { (Rc(obj1))(k,eta,obj2.amount,Ff(obj1))
      (k/eta*obj2.amount*Ff(obj1)) }
    alg_infl { (Cf(obj1))(Rc(obj1),NA,obj1.A)(Rc(obj1)/NA*obj1.A) }
    aggr_infl { (distr_neutron(obj1)[i])(Rc(obj1),pdf_neutron)
      (Rc(obj1)*pdf_neutron[i],i=0..8) }
    alg_infl { (pn(obj1))(distr_neutron(obj1))
      (sum(i*distr_neutron(obj1)[i],i=0..8)) }
    aggr_infl { (distr_A1(obj1)[i])(Rc(obj1),pdf_A1)
      (Rc(obj1)*pdf_A1[i],i=73..117) }
```

Figure 4.6: Aggregate effects model of fission.

```
        aggr_infl { (distr_A2(obj1)[i])(Rc(obj1),pdf_A2)
            (Rc(obj1)*pdf_A2[i],i=118..162) }
        aggr_infl { (distr_E_neutron(obj1)[i])(pn(obj1),pdf_E_neutron)
            (pn(obj1)*pdf_E_neutron[i],i=0..8) }
        aggr_infl { (distr_E_gamma(obj1)[i])(Rc(obj1),pdf_E_gamma)
            (Rc(obj1)*pdf_E_gamma[i],i=0..8) }
        aggr_infl { (distr_E_F1(obj1)[i])(Rc(obj1),pdf_E_F1)
            (Rc(obj1)*pdf_E_F1[i],i=40..84) }
        aggr_infl { (distr_E_F2(obj1)[i])(Rc(obj1),pdf_E_F2)
            (Rc(obj1)*pdf_E_F2[i],i=85..129) }
        cum_infl { (Fissile_fissioned)(Rc(obj1))(Rc(obj1)) }
        cum_infl { (Fissile_consumed)(Cf(obj1))(Cf(obj1)) }
    }
    products {
        cum_infl { (obj1.amount)(Cf(obj1))(-Cf(obj1)) }
        cum_infl { (obj2.amount)(Rc(obj1))(-Rc(obj1)) }
        cum_infl { (Neutron.amount)(pn(obj1))(pn(obj1)) }
        cum_infl { (Neutron.energy)(distr_E_neutron(obj1)) (sum((i+ei/2)*
            distr_E_neutron(obj1)[i],i=0..8)) }
        cum_infl { (Gamma.energy)(distr_E_gamma(obj1))
            (sum((i+ei/2)*distr_E_gamma(obj1)[i],i=0..8)) }
        distr_infl { (F1)(distr_A1,pdf_dZ)
            (F1.Z=c_n.Z/(c_n.A-nu)*A+dZ;
            F1.N=A-(c_n.Z/(c_n.A-nu)*A+dZ);
            F1.amount=distr_A1(obj1)[A]*pdf_dZ[dZ]/NA*A, dZ=-4..4,A=73..117) }
        distr_infl { (F2)(distr_A2,pdf_dZ)
            (F2.Z=c_n.Z/(c_n.A-nu)*A+dZ;
            F2.N=A-(c_n.Z/(c_n.A-nu)*A+dZ);
            F2.amount=distr_A2(obj1)[A]*pdf_dZ[dZ]/NA*A,dZ=-4..4,A=118..162) }
        cum_infl { (Energy_F1.amount)(distr_E_F1(obj1))
            (sum((i+ei/2)*distr_E_F1(obj1)[i],i=40..84)) }
        cum_infl { (Energy_F2.amount)(distr_E_F2(obj1))
            (sum((i+ei/2)*distr_E_F2(obj1)[i],i=85..129)) }
    }
}
```

Figure 4.7: Continuation of aggregate effects model of fission.

```
aggregate_effects_model radioactive_capture (rc)
{
   material { obj1 }
   activity_conditions { obj1.amount > 0 & obj2.amount > 0 }
   parametric_reaction {
      def_const NA (value: 6.02217e+23) }
   relations {
      alg_infl { (Sigma_a_fuel)(obj1.amount,obj1.sigma_g)
         (obj1.amount*obj1.sigma_g) }
      alg_infl { (Fg(obj1))(obj1.amount,obj1.sigma_g,Sigma_a_fuel)
         (obj1.amount*obj1.sigma_g/Sigma_a_fuel) }
      alg_infl { (nrc(obj1))(obj2.amount,k,eta,Fg(obj1))
         (obj2.amount*k/eta*Fg(obj1)) }
      alg_infl { (Prc(obj1))(nrc(obj1),obj1.A,NA)(nrc(obj1)*obj1.A/NA) }
      alg_infl { (Crc(obj1))(nrc(obj1),obj1.A,NA)(nrc(obj1)*obj1.A/NA) }
      aggr_infl { (distr_E_gamma(obj1)[i])(nrc(obj1),pdf_E_gamma)
         (nrc(obj1)*pdf_E_gamma[i],i=0..8) }
   }
   products {
      cum_infl { (obj1.amount)(Crc(obj1))(-Crc(obj1)) }
      cum_infl { (obj2.amount)(nrc(obj1))(-nrc(obj1)) }
      cum_infl { (P.amount)(Prc(obj1))(Prc(obj1)) }
      cum_infl { (Gamma.energy)(distr_E_gamma(obj1))
         (sum((i+ei/2)*distr_E_gamma(obj1)[i],i=0..8)) }
   }
}
```

Figure 4.8: Aggregate effects model of radioactive capture.

```
aggregate_effects_model beta_negative_decay(bnd)
{
    material { obj1 }
    activity_conditions { obj1.amount > 0 }
    parametric_reactions { def_const ld (value:0.693) }
    relations {
        decay_infl { (obj1.dmode & BETA_POSITIVE)(Pbnd(obj1))
            (ld,obj1.halflife,obj1.amount,NA,obj1.A,cycle_time)
            (0.5*ld*obj1.halflife*obj1.amount*NA/obj1.A*cycle_time) }
        decay_infl { (obj1.dmode & BETA_POSITIVE)(Cbnd(obj1))
            (ld,obj1.halflife,obj1.amount,NA,obj1.A,cycle_time)
            (0.5*ld*obj1.halflife*obj1.amount*NA/obj1.A*cycle_time) }
        decay_infl { (!(obj1.dmode & BETA_POSITIVE))(Pbnd(obj1))
            (ld,obj1.halflife,obj1.amount,NA,obj1.A,cycle_time)
            (ld*obj1.halflife*obj1.amount*NA/obj1.A*cycle_time) }
        decay_infl { (!(obj1.dmode & BETA_POSITIVE))(Cbnd(obj1))
            (ld,obj1.halflife,obj1.amount,NA,obj1.A,cycle_time)
            (ld*obj1.halflife*obj1.amount*NA/obj1.A*cycle_time) }
    }
    products {
        cum_infl { (obj1.amount)(Cbnd(obj1),NA,obj1.A)(-Cbnd(obj1)/NA*obj1.A) }
        cum_infl { (P.amount)(Pbnd(obj1),NA,obj1.A)(Pbnd(obj1)/NA*obj1.A) }
        cum_infl { (Beta_Neg.amount)(Pbnd(obj1))(Pbnd(obj1)) }
        cum_infl { (Beta_Neg.energy)(Pbnd(obj1))(0.4*Pbnd(obj1)) }
    }
}
```

Figure 4.9: Aggregate effects model of beta negative decay.

```
aggregate_effects_model beta_positive_decay(bpd) {
    material { obj1 }
    parametric_reactions {
        def_const ld (value: 0.693) }
    activity_conditions { obj1.amount > 0 }
    relations {
        decay_infl { (obj1.dmode & BETA_NEGATIVE)(Pbpd(obj1))
            (ld,obj1.halflife, obj1.amount,NA,obj1.A,cycle_time)
            (0.5*ld*obj1.halflife*obj1.amount*NA/obj1.A*cycle_time) }
        decay_infl { (obj1.dmode & BETA_NEGATIVE)(Cbpd(obj1))
            (ld,obj1.halflife, obj1.amount,NA,obj1.A cycle_time)
            (0.5*ld*obj1.halflife*obj1.amount*NA/obj1.A*cycle_time) }
        decay_infl { (not(obj1.dmode & BETA_NEGATIVE))(Pbpd(obj1))
            (ld,obj1.halflife, obj1.amount,NA,obj1.A,cycle_time)
            (ld*obj1.halflife*obj1.amount*NA/obj1.A*cycle_time) }
        decay_infl { (not(obj1.dmode & BETA_NEGATIVE))(Cbpd(obj1))
            (ld,obj1.halflife, obj1.amount,NA,obj1.A,cycle_time)
            (ld*obj1.halflife*obj1.amount*NA/obj1.A*cycle_time) }
    }
    products {
        cum_infl { (obj1.amount)(Cbpd(obj1),NA,obj1.A)(-Cbpd(obj1)/NA*obj1.A) }
        cum_infl { (P.amount)(Pbpd(obj1),NA,obj1.A)(Pbpd(obj1)/NA*obj1.A) }
        cum_infl { (Beta_Pos.amount)(Pbpd(obj1),obj1.A)(Pbpd(obj1)) }
        cum_infl { (Beta_Pos.energy)(Pbpd(obj1),obj1.A)(0.4*Pbpd(obj1)) }
    }
}
```

Figure 4.10: Aggregate effects model for beta positive decay.

Any term "obj1" or "(obj1)" in the aggregate effects model will be associated with the identity of the object that instantiates the aggregate effects model. For example, in the case of the radioactive capture process, when ^{235}U is active, and instantiates the basic nuclear physics model of radioactive capture, it, in turn, instantiates the corresponding aggregate effects model. NPT associates Prc(obj1) with PrcU235 (see Fig. 4.8), which has the semantic meaning of a product of radioactive capture process instantiated by ^{235}U. The amount of PrcU235 is influenced by the number of radioactive capture processes nrcU235, the mass number of ^{235}U (U235.A), and Avogadro's number NA. This amount is expressed through the algebraic influence **alg_infl** { Prc(obj1) (nrc(obj1),obj1.A, NA) (nrc(obj1) * obj1.A/NA) } in Fig. 4.8. The **alg_infl** is discussed in Section 4.4.

Each type of active nuclide (e.g. U235) which actives the BNPM of certain process (e.g. radioactive capture process) will determine the type of products nuclides P through P.Z and P.N defined in BNPM (see Fig. 4.8). Knowing values of P.Z and P.N, the type of product nuclides can be known directly from the representation of nuclide data. The result, then, will be used to replace any term P specified in the AEM (e.g. P.amount) to produce a state space equation showing the amount of product nuclides at the time when the certain type of nuclide is active.

4.2 Symbolic Level

The symbolic level consists of objects, their properties, and relations which are defined in an input description. The symbolic level also consists of objects representing instances of the basic nuclear physics model (BNPM) definitions and aggregate effects model (AEM) definitions. An example of an input description is shown in Appendix II.

Whenever one refers to BNPM and AEM, they exist in one of three possible stages: definition stages, instantiation stages, and activity stages. At the definition

stage, BNPM and AEM are purely an abstraction of the nuclear physics processes, and are not related to any actual objects.

At the instantiation stage, NPT identifies a subset of objects which, when assigned to reactants of the basic nuclear physics model, satisfy the reactant fields. The system also identifies every basic nuclear physics model which, when assigned to the aggregate effects model, satisfies the aggregate effects model conditions.

For each set of objects which is bound to the reactants of the BNPM, instances of the BNPM are created. Some of these instances are then bound to the AEM which in turn instantiates the AEM. For example, consider a set of objects for a reactor core simulation fueled by a mixture of U235 and U238 such as defined in the input descriptions shown in Appendix II.

These objects are considered to be active. The subset of instantiations of these objects, U235 and Neutron are bound to reactants of the basic nuclear physics model of fission and radioactive capture. This causes the instances of the basic nuclear physics models, BNPM fission (U235, Neutron) and BNPM radioactive capture (U235, Neutron), to be created. Similar actions take place for U238. These instantiations, along with some instantiations of the basic objects PHWNUR,HW,Zra are bound to the aggregate effects models of fission. This causes the instantiations of the aggregate effects models, AEM (BNPM fission(U235,neutron),PHWNUR,HW,Zra) and AEM (BNPM radioactive_capture(U235,neutron)). Fig. 4.11 shows an example of instances of objects along with instances of BNPM and AEM for radioactive capture and fission when U235 and U238 instantiate these models.

These instances are considered as conceivable processes. Some of these conceivable processes are not active at the activating stage, which is discussed later.

```
U235
neutron
PHWNUR
HW
Zra
BNPM radioactive_capture(U235,neutron)
AEM (BNPM radioactive_capture(U235,neutron))
BNPM fission(U235,neutron)
AEM (BNPM fission(U235,neutron),PHWNUR,HW,Zra)
U238
BNPM radioactive_capture(U238,neutron)
AEM (BNPM radioactive_capture(U238,neutron))
BNPM fission(U238,neutron)
AEM (BNPM fission(U238,neutron),PHWNUR,HW,Zra)
```

Figure 4.11: Example of a set of conceivable processes at the end of the first cycle.

4.3 Quantitative Level

An instance of the basic nuclear physics model and aggregate effects model does not cause any analysis of the system. Only active instances can cause the analysis. After instantiations of BNPMs and AEMs, the next step is to identify the active instances.

Active instances are obtained when objects are bound to reactants, the instantiated basic nuclear physics model is bound to the aggregate effects model, and the parameters of the objects satisfy activity conditions for both the BNPM and the AEM, and any preconditions of the AEM. The basic objects are always considered active, i.e. U235, U238, Neutron, PHWNUR, HW, Zra. The active processes which come from Fig. 4.11 are shown in Fig. 4.12. The last two lines in Fig. 4.11 do not appear as active processes. This is due the fact that when U238 is active, its fission cross section, U238.sigma_f, does not satisfy the activity condition of a fission process. Therefore the BNPM of fission is not activated, which, in turn, means that the AEM of the fission process for U238 is not activated.

The set of active BNPMs and AEMs are set to follow nuclear process phenomena

```
U235
neutron
PHWNUR
HW
Zra
BNPM radioactive_capture(U235,neutron)
AEM (BNPM radioactive_capture(U235,neutron))
BNPM fission(U235,neutron)
AEM (BNPM fission(U235,neutron),PHWNUR,HW,Zra)
U238
BNPM radioactive_capture(U238,neutron)
AEM (BNPM radioactive_capture(U238,neutron))
```

Figure 4.12: Active processes at the end of the first cycle of the experiment.

in the current state. At the activating stage, the interactions between nuclear physics processes $Q = q_1, q_2 \cdots q_m$ take place. Several tasks are carried out as follows:

1. Each active nuclide T_a (possibly from the product nuclides of the previous cycle) and possible particles (beta decay processes do not need particles) generates state space and probabilistic space equations SP for active processes Q, i.e. $SP(q_1, T_a, [t_i, t_{i+1}]) \cdots SP(q_m, T_a, [t_i, t_{i+1}])$. The state space equations are used to answer the following question: how will the state of the system evolve over the next cycle period given the current state? The probabilistic space equations are used to answer questions such as: what is the distribution of products (e.g. neutrons or fission fragments) in the current state?

2. The type of product nuclides for each active process, (e.g. P for the radioactive capture process, see Fig. 4.8), is characterized by its atomic number P.Z and its neutron number P.N specified in the product field of the BNPM. The type of product particle, (e.g. Beta_Pos for the beta positive decay process, see Fig. 4.4), is characterized by its particle identity Beta_Pos.id specified in the product field of the BNPM.

The atomic and the neutron numbers of product nuclides are dependent on the type of activating nuclide (e.g. obj1). Knowing the values of P.N and P.Z, the behaviour generator of NPT (discussed later) determines the type of product nuclides P using the representation of nuclide data [18].

3. NPT substitutes any symbolic name defined in the BNPM and AEM, (e.g. "(obj1)" or "obj1"), with the actual identity of the activating nuclides. This gives NPT the capability of instantiating BNPMs and AEMs by different types of activating nuclides, which in turn, produces the correct type of product nuclides and their attributes required for calculating the amount of product nuclides, as well as establishing the appropriate energy distribution. NPT substitutes any symbolic name defined in the AEM as "F1" or "F2", with an actual nuclide name from the representation of nuclide data [18]. Once the type of the product nuclides for current active process is known, NPT substitutes any symbolic name of product nuclides P defined in the AEM with actual name of product nuclides.

4. If no more nuclides are active at the current cycle, the state space equations then are evaluated. The state space equations predict how much material is consumed, how much remains as fuel along with its composition, and how much energy is released from product particles and from product nuclides.

4.4 Influences in NPT

Besides characterizing nuclear processes, NPT describes the connections between variables. The connections between variables, i.e. how one or a set of variables affect another variable or another set of variables, are expressed in terms of influences. There are several types of influences in NPT which are described here. The formal syntax for all NPT influences is given in Appendix III.

Algebraic influences specify how the value of a given variable is affected by one or more variables. The syntax for the algebraic influence is given using a context-free grammar [39] as follows:

```
<alg_infl>          ::= "alg_infl" "{" "(" <infl_ed_var> ")"
                        "(" <list_infl_ing_vars> ")"
                        "(" <funct_spec> ")"  "}"
```

The semantic interpretation for the algebraic influences is that an algebraically influenced variable equals the sum of the function specifications specified in the algebraic influences of active processes affecting that variable.

For example, the macroscopic absorption cross section, Sigma_a_fuel,, is affected by the fission process and by the radioactive capture process. From the fission process, Sigma_a_fuel, is affected by the amount of activating nuclides obj1.amount, and the fission cross section obj1.sigma_f, through function obj1.amount, * obj1.sigma_f. From the radioactive capture process, Sigma_a_fuel, is influenced by the amount of activating nuclides obj1.amount, and the capture cross section obj1.sigma_g through function obj1.amount, *obj1.sigma_g. These influences can be seen in Figures 4.6 and 4.8.

An aggregate influence specifies how the values of a vector variable are influenced by vector variable(s) or by variable(s) and vector variable(s) through specific operations. The syntax of this influence is as follows:

```
<aggr_infl>          ::= "aggr_infl" "{" "(" <infl_ed_vect_var> ")"
                         "(" <infl_ing_vects_vars_vects> ")"
                         "(" <func_spec> ")" "}"
<infl_ing_vects_vars_vects> ::=  <infl_ing_vect_vars> |
                          <infl_ing_var_and_vect_vars>
```

The semantic interpretation for the aggregate influences is that an aggregately influenced vector variable equals the vector operation specified in the aggregate influence affecting that vector variable.

For example, when the fission process is active as a result of fissioning a type of nuclide, the number of fissions per cycle influences the distribution of fission neutrons produced as specified by distr_neutron. This distribution influences the number of fission neutrons produced. This is stated as

```
aggr_infl { (pn(j))(distr_neutron(j)) (sum(i*distr_neutron(j)[i], i=0..8)) }
```

where a Poisson distribution is specified in this case. Complete details of this example are given in Appendix I.1.7.1.

A cumulative influence specifies how the value of a variable is affected cumulatively by one variable, or by an operation on a set of influencing variables, or by an operation on a set of elements of a vector variable. The syntax of this influence is as follows:

```
<cum_infl>          ::= "cum_infl" "{" "(" <infl_ed_var> ")"
                        "(" <list_infl_ing_vars> ")"
                        "(" <funct_spec> ")"  "}"
```

The semantic interpretation for the cumulative influences is that they specify how cumulatively influenced variables are influenced by the influencing variables.

Unlike **alg_infl**, which assigns the value of the expression to the influenced variable, **cum_infl** accumulates the result of evaluating the function. For example, the amount of material is influenced by processes that are active (e.g. fission and radioactive capture). The amount of material consumed for fission process Cf(obj1) decreases the amount of fissile material obj1.amount. This is stated as

```
cum_infl { (obj1.amount)(Cf(obj1))(-Cf(obj1)) }
```

Similarly, the amount of material consumed for radioactive capture decreases obj1.amount. The function specification can be seen in Fig. 4.8.

A distribution influence specifies how the values of a set of NPT output objects are influenced by vector variable(s) or variable(s) and vector variable(s) through specific repeated operations. The syntax of this influence is given by

```
<distr_infl>        ::= "distr_infl" "{" "(" <out_obj_symb> ")"
                        "(" <infl_ing_vects_vars_vects>  ")"
                        "(" <loop_expr> ")" "}"
<loop_expr>         ::= <in_loop_expr>+  <loop_counter> <more_loop>*
<more_loop>         ::= "," <loop_counter>
<in_loop_expr>      ::= <out_obj_ident> "=" <func_spec> ";"
```

For distribution influences, the semantic interpretation is that a set of influenced NPT object attributes equals operations between a set of vector variables specified in the distribution influence affecting that attribute. An example of distribution influences for fission (see also Fig. 4.7) is as follows:

distr_infl { (F1)(distr_A1,pdf_dZ)

(F1.Z=c_n.Z/(c_n.A-nu)*A+dZ;

F1.N=A-(c_n.Z/(c_n.A-nu)*A+dZ);

F1.amount=distr_A1(obj1)[A]*pdf_dZ[dZ]/NA*A;

dZ=-4..4,A=73..117) }

The terms obj1 and distr_A1(**obj1**) are used to refer the type of fissioning nuclide (e.g. ^{235}U) and its mass number distributions, respectively. The details are discussed in

Appendix I.1.7.2.

A decay influence specifies how a variable is influenced by a set of variables through certain operations when conditions for a decay process are fulfilled. The syntax of this influence is

```
<decay_infl>        ::= "decay_infl" "{" "(" <cond_expr> ")"
                        "(" <infl_ed_var> ")" "(" <infl_ing_vars> ")"
                         <infl_expr> "}"
```

For decay influences, the semantic interpretation is that an influenced variable caused by decay processes is equal to the the sum of the expressions specified in the decay influences of decay processes affecting that variable. An example of a decay influence is as follows:

decay_infl { (not(obj.dmode & BETA_NEGATIVE))

(Pbpd(obj))(obj.halflife, obj.amount, cycle_time)

(ld / obj.halflife * obj.amount * cycle_time) }

This shows that when the decay mode of the activating object (nuclide) for the beta positive decay is BETA POSITIVE, not BETA NEGATIVE, then the product nuclides Pbpd(obj) are influenced by the nuclide's halflife obj.halflife, the number of the activating nuclides obj.amount and the cycle time cycle_time. The influence is evaluated using the function specification
ld / obj.halflife * obj.amount * cycle_time. The value of the constant ld is 0.693 [21]. The term ld/obj.halflife represents a decay constant of the activating nuclides.

4.5 Deriving State Space and Probabilistic Space Equations

4.5.1 Algebraic Influences

State space and probabilistic space equations are used to answer questions such as: how will the state of the system evolve over the next iteration given the current initial state? Both state space equations and probabilistic space equations are established by extracting all influences; algebraic, aggregate, and cumulative influences, and combining them. The influences are specified in the AEMs. Algebraic influences are designed to express dependency between an influenced variable and influencing variables which may come either from different types of processes or the same processes that are activated by different types of nuclides. Algebraic influences expressed in different AEMs may have the same influenced variables. The influenced variable is established by performing an addition operation of all function specifications of the influenced variables. In addition, an object identity is attached to any variable having notation (obj1) in order to recognize which type of nuclides activate the processes.

For example, the first influence in the AEM of radioactive capture tells NPT how to calculate macroscopic absorption cross section, Sigma_a [21]. When ^{235}U activates the AEM of radioactive capture, NPT generates

$$Sigma\_a = U235.amount * U235.sigma\_g. \tag{4.4}$$

Later, ^{238}U activates the AEM of radioactive capture, and ^{238}U contributes a term

$$U238.amount * U238.sigma\_g$$

to Sigma_a, and this leads to

$$Sigma\_a = U235.amount * U235.sigma\_g + U238.amount * U238.sigma\_g.$$

4.5.2 Aggregate Influences

Unlike the algebraic influences, aggregate influences are established, not combined, by performing operations expressed in their function specifications. This is due to the fact that the aggregate influence is designed for expressing dependency between non-deterministic quantities represented as a vector (or quantities stored in a vector) and deterministic quantities. The aggregate influence shown in Fig. 4.8 tells NPT to fill the first 9 elements of distr_E_gammaU235 with the products of nrcU235 and the first 9 elements of pdf_E_gammaU235 when ^{235}U instantiates the aggregate effects model of radioactive capture. The probabilistic space equation is generated as follows:

$$distr\_E\_gammaU235 = nrcU235 * pdf\_E\_gammaU235$$

The probabilistic space equation of distr_E_gammaU238 is generated in the same manner.

4.5.3 Cumulative Influences

Cumulative influences are designed to express dependency between influenced and influencing variables cumulatively. Like aggregate influences, all cumulative influences are established, not combined, by performing operations expressed in their function specifications. Unlike other influences which replace the value of the variable on the left hand side of the equation with the value of the right hand side of the equation, cum_infl decreases, increases, multiplies, or divides the value of the variable on the left hand side of the equation with the value of the right hand side of the equation. This influence is very useful for accumulating the amount of NPT objects. For example, the first cum_infl in Fig. 4.8 tells NPT to decrease the amount of ^{235}U with the amount of ^{235}U consumed for radioactive capture represented as CrcU235 when ^{235}U instantiates the AEM for radioactive capture. The generated probabilistic state space

equation will be

$$U235.amount + = - CrcU235$$

The same applies to ^{238}U.

4.5.4 Decay Influences

Decay influences are designed to express dependency between the influenced variable and influencing variables when decay processes occur. A nuclide may undergo more than one decay process depending on its decay modes. If more than one decay process occurs, the product nuclides come from different decay processes. In this case, the number of product nuclides (the influenced variable) influenced by many decay processes equal to the sum of the function specifications specified in the decay processes affecting the variable. Since a nuclide can have more than one decay mode, each type of nuclide possibly activates two decay processes β^+ and β^-. In this case, when the nuclide instantiates a β^+ decay process, it produces a state space equation for product nuclides

$$0.5 * 0.693 * obj1.halflife * obj1.amount \tag{4.5}$$

where obj1 will be replaced with the nuclide identity of the instantiating nuclide. The same state space equation is produced from the β^- decay process. The factor 0.5 in equation (4.5) indicates that half of the product nuclides for the decay processes come from the β^+ decay process, and the rest from the β^- decay process. If the type of nuclide activates only one type of decay process, then the state space equation

$$0.693 * obj1.halflife * obj1.amount$$

is produced from the decay process, where obj1 will be replaced with the nuclide identity of the instantiating nuclide.

4.5.5 Distribution Influences

Distribution influences are designed to express dependencies between a set of NPT objects with the appropriate distributions affecting the objects. For example, when a type of nuclide activates a fission process, two sets of fission fragments, symbolically represented by F1 and F2, are produced. The mass numbers of the two sets of fission fragments F1 and F2 are probabilistic. The distribution of the mass numbers of the fission fragments are given by distr_A1 for F1 and distr_A2 for F2. For F1, the **distr_infl** specified in the AEM of an active fission process is

```
distr_infl { (F1)(distr_A1,pdf_dZ)
  (F1.Z=c_n.Z/(c_n.A-nu)*A+dZ;
  F1.N=A-(c_n.Z/(c_n.A-nu)*A+dZ);
  F1.amount=distr_A1(obj1)[A]*pdf_dZ[dZ]/NA*A;
  dZ=-4..4,A=73..117) }
```

Having this **distr_infl**, the probabilistic space equation for each fission fragment of F1 will be produced showing the type of product nuclides produced, and their amount. This is carried out by the behaviour generator of NPT, explained in section 5.2, by the following sequence of steps:

1. Determine its atomic number F1.Z and its neutron number F1.N using the first two expressions in the **distr_infl**.

2. Determine the actual nuclide identity for F1.Z and F1.N using the representation of nuclide data [18].

3. Generate a probabilistic space equation, which shows the number of nuclides having F1.Z and F1.N, by replacing the F1, A, dZ of equation (4.6) with the actual nuclide identity, the appropriate A and the appropriate dZ, respectively.

$$F1.amount = distr\_A1(obj1)[A] * pdf\_dZ[dZ]/NA * A. \qquad (4.6)$$

4. This sequences of steps is carried out for the values of dZ from -4 to 4, and A from 73 to 117.

For example, when a type of nuclide ^{235}U instantiates the fission process, having **distr_infl** for F1 specified in the AEM (see Fig. 4.6), the first expression of **distr_infl**, F1.Z=c_n.Z/(c_n.A-nu)*A+dZ , gives the atomic number of fission fragment F1. By replacing c_n.Z, c_n.A, A, and dZ with the values of U236.Z, U236.A, one of values in a range [73,117], and one of values in a range in [-4,4], the atomic number F1.Z of a fission fragment is obtained. The second expression of **distr_infl**, F1.N=c_n.Z/(c_n.A-nu)*A+dZ , gives the corresponding neutron numbers of the fission fragment F1. From the obtained values of F1.Z and F1.N, the type of fission fragment is known from the representation of the nuclide data [18]. The third expression, F1.amount=distr_A1(obj1)[A]*pdf_dZ[dZ]/NA*A , produces a probabilistic space equation showing the number of nuclides of fission fragment F1.

As an example, when the mass number A is 90, and dZ = 1, the atomic number of fission fragment F1.Z = 36 is obtained. The corresponding neutron number F1.N = 54 is obtained. The actual type of nuclide having F1.Z = 36 and F1.N = 54 is found to be Kr90. Therefore, one of the probalistic space equations generated by the NPT generator behaviour is

$$Kr90.amount = distr\_A1U235[90] * pdf\_dZ[1].$$

Analogously, other probabilistic space equations for other types of fission fragments are produced. An example of the fission process producing different types of fission fragments is discussed in section 6.1.

The complete set of equations generated from the relations and product definitions in the AEMs of all processes usually changes from cycle to cycle depending upon the number of types of activating nuclides.

After equations are generated, the behaviour generator of NPT evaluates the state space and probabilistic space equations to obtain quantitative values. As a consequence, new objects may be born or existing objects may disappear every cycle time, which leads to a dynamic state space and probabilistic state space description.

Chapter 5

Implementation

This chapter discusses the design of a nuclear process simulation tool and the implementation of the NPT models.

5.1 Design and architecture of a nuclear process simulation tool

A nuclear process simulation tool can be used, for example, to simulate nuclear physics processes in a reactor core (see Fig. 5.1). The NPTsim simulation tool enables the study of, and experimentation with, nuclear process interactions by way of the NPT symbolic representation. By changing simulation parameters and observing the resulting output, the behaviour of the specific nuclear physics processes can be observed.

The simulation kernel is composed of the internal representation of NPTsim where the functional model of NPTsim is shown in Fig. 5.4. The model description file contains the knowledge level components discussed before; i.e. the basic nuclear

physics models and the aggregate effects models of different types of processes (fission, radioactive capture, beta negative decay and beta positive decay).

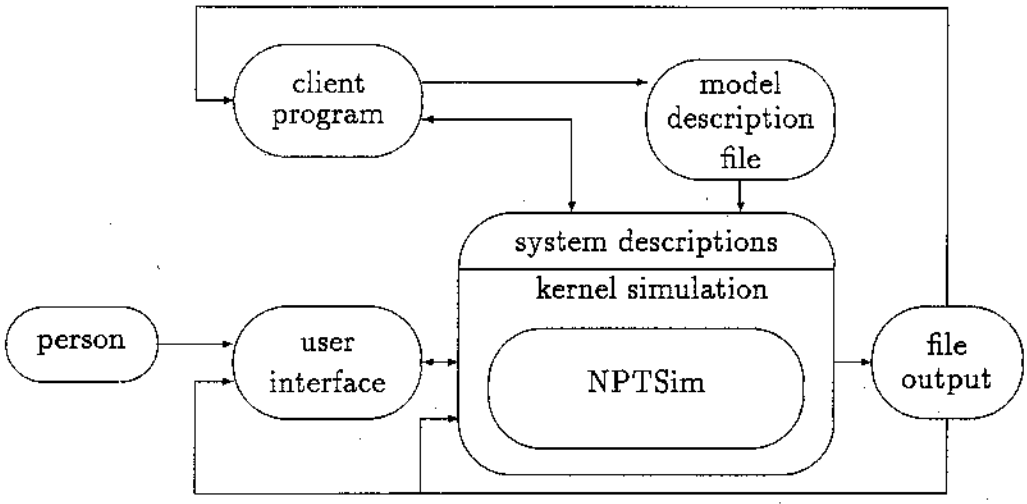

Figure 5.1: Components of a nuclear process simulation tool.

Fig. 5.2 shows the high-level activities of simulating nuclear processes in a cycle. Once the basic nuclear physics and aggregate effects models are included in the kernel, NPT identifies all possible instantiations of the basic nuclear physics and aggregate effects models. This task is accomplished in two steps. First, the instantiations of the basic nuclear physics model for each type of process are determined. Second, the instantiations of the corresponding aggregate effects models using the results of instantiation of basic nuclear physics models are determined.

A basic nuclear physics model of a type of process is instantiated when the nuclide objects and their conditions, as well as their attributes, match the requirements for the individual reactants, and reactant conditions as stated in the basic nuclear physics models. These requirements are symbolically represented as $C1$.

The results of the instantiation of basic nuclear physics models, along with reactor parameters stated in the input description, are used to instantiate the corresponding

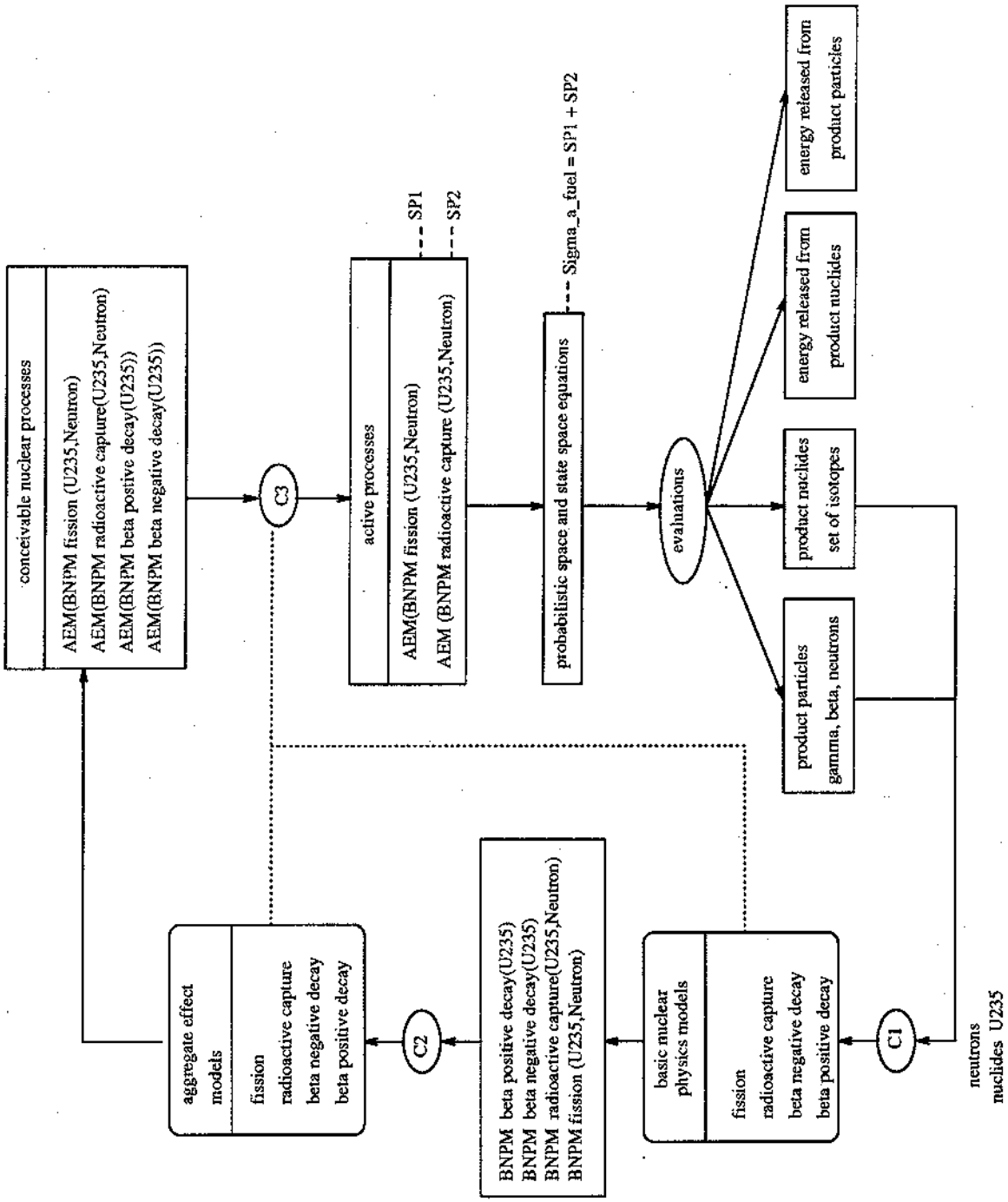

Figure 5.2: High-level activities of simulating nuclear processes in a cycle.

aggregate effects models. An aggregate effects model is instantiated when the requirements for the reactants, material, and equipment stated in the aggregate effects model are satisfied. These requirements are symbolically represented as $C2$. The results of the instantiation of basic nuclear physics and aggregate effects models are considered as conceivable nuclear processes.

The program proceeds to test the validity of the preconditions and activity conditions for each instance. The activity conditions are specified in both the basic nuclear physics and aggregate effects models, which are shown in Fig. 5.2 by the dotted lines. The preconditions and the activity conditions are symbolically represented as $C3$. Testing the validity of both preconditions and activity conditions, the program requires that the initial values for at least a subset of parameters and variables have been specified. The result of this test is a set of active processes, which generates state space and probabilistic space equations showing the interactions among objects (particles and nuclides) existing in the reactor core. Fig. 5.2 shows an example equation Sigma_a_fuel resulting from interactions between neutrons and nuclides through radioactive capture and fission processes. The actual expression containing $SP1$ and $SP2$ for the first example can be seen in this example. When U235 activates a fission process, the NPT behaviour generator produces Sigma_a_fuel = U235.amount * U235.sigma_f which is symbolically written as $SP1$ in Fig. 5.2. Similarly, when U235 activates a radioactive capture process, the NPT behaviour generator produces Sigma_a_fuel = U235.amount * U235.sigma_g which is symbolically written as $SP2$ in Fig. 5.2. The final state space equation for Sigma_a_fuel when U235 activates two types of processes is

$$Sigma\_a\_fuel = U235.amount * U235.sigma\_f + U235.amount * U235.sigma\_g.$$

Once the state space and probabilistic space equations have been generated, and no more nuclides are active in the current cycle, the NPT interpreter evaluates the equations to produce quantities of products for the nuclear processes involved. The

products include product particle(s), product nuclide(s), energy of product particle(s) released and energy release of product nuclide(s), along with the amount of fuel consumed.

When a cycle is completed, there are possibly more types of nuclides available in the reactor core. Each type of nuclide may activate one or more processes that are available in the system depending on their properties. Each type of nuclide will activate its associated basic nuclear physics and aggregate effects models again as shown in Fig. 5.2. This process is repeated in a loop. The conceivable nuclear processes and the active processes as well as the state space equations generated in the next cycle may be different from those generated in the current cycle. The changes depend on how many types of nuclides are available in the system and their properties (e.g. fission cross section, halflife).

The products of nuclear reactions every cycle time are recorded in the system description as shown in Fig. 5.1. The products (e.g. neutron products, product nuclides) are fed back into the simulation kernel to do same procedures described above and to generate other sets of products for the next cycle time.

The system description also contains an initial set of parameters of the objects involved in nuclear processes, along with their starting amount, runtime of simulation, report interval required during the simulation, and cycle time (e.g. neutron cycle time).

Once the runtime, the report interval, and cycle time are set in the system description, simulation is performed and the behaviour of the system can be obtained. For a nuclear fission process, for example, the system predicts how much material is consumed, how much material remains as fuel, the quantity of fission products generated along with their composition (including neutron products), how much thermal energy is released from both fission products and neutron products, and how much energy is generated by the resulting gamma rays.

The above products are reported in an output file at every report interval time

step of the simulation. Through a simple user interface, a user can interact directly with the kernel simulation in order to change the parameters of the nuclear processes being modeled.

In NPT, each type of nuclide is capable of carrying out processes in the same manner as described in Chapter 3. When a type of nuclide interacts with neutrons through a specific process, the associated actions in the NPT model have active processes from instances of the basic nuclear physics and the corresponding aggregate effects models.

For example, in the case of Th233, it will instantiate the basic nuclear physics models of fission, radioactive capture and beta negative decay and the corresponding aggregate effects models which result in the active processes of fission, radioactive capture and beta negative decay.

NPT uses basic nuclear physics models and aggregate effects models as a modeling language for each type of nuclear process. The NPT behaviour generator can generate state space equations and probabilistic space equations every cycle time (or neutron-life time in this example) from active processes. Both the state space and probabilistic space equations show the interactions between objects (for example neutrons and nuclides) which already existed in the reactor core and which become available as a result of nuclear processes up to a particular time period.

Consider Fig. 5.3. Assuming that all relevant basic nuclear physics and aggregate effects models are included in the system (basic nuclear physics and aggregate effects models of fission, radioactive capture, beta positive and beta negative decays), the NPT behaviour generator identifies all possible instantiations of the basic nuclear physics and aggregate effects models. This task is accomplished in two steps, first determining instantiations of the basic nuclear physics model for each type of process, and then determining the corresponding instantiation of aggregate effects models using the results of instantiation of basic nuclear physics models.

A basic nuclear physics model of a process is instantiated when a set of NPT

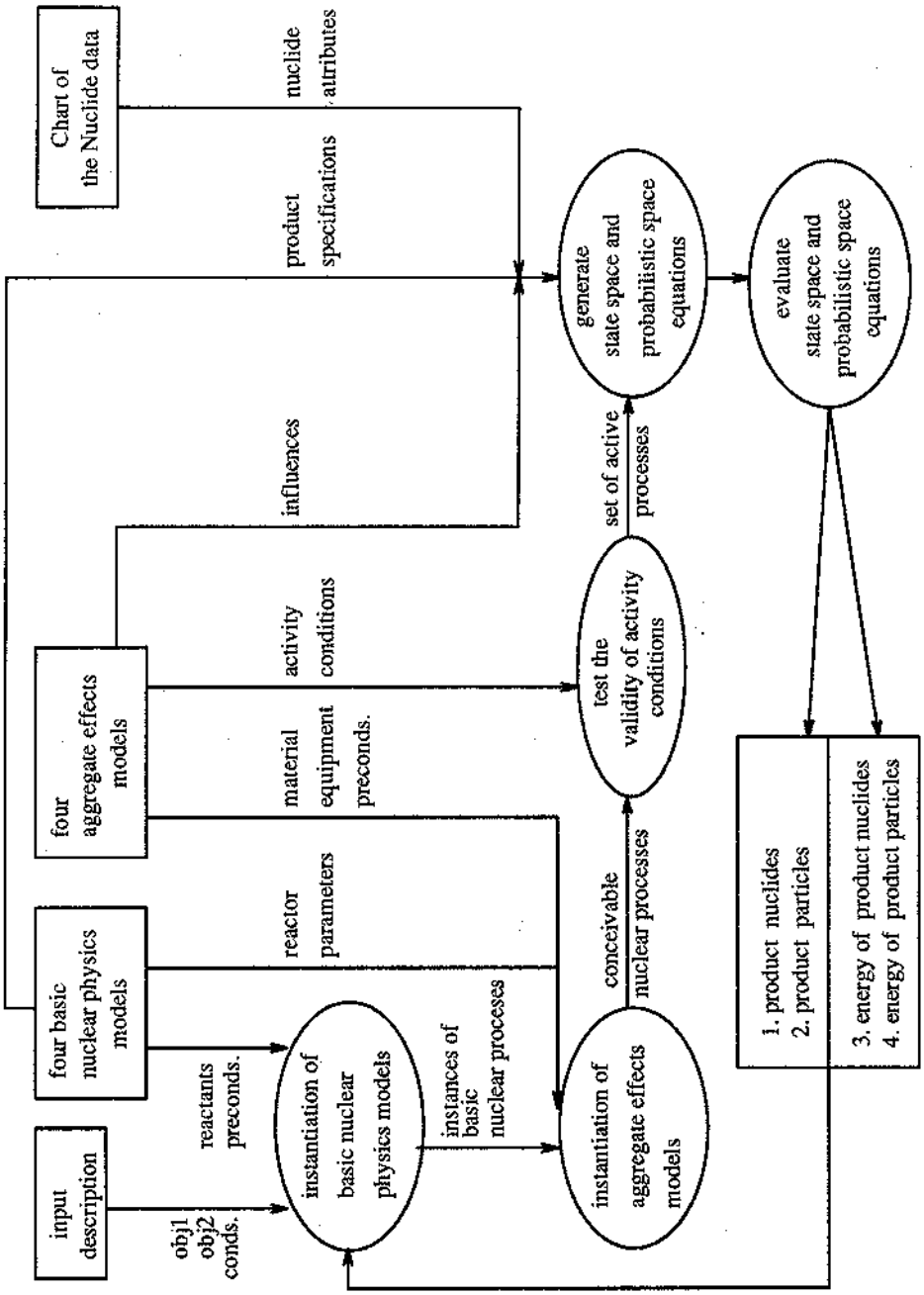

Figure 5.3: High-level description of simulation using the NPT model.

objects match the requirements for the individual reactants as defined in the basic nuclear physics models. The high-level process showing the flow of data values from their sources to their destination for every neutron life-time or cycle time is shown in Fig. 5.3.

The results of the instantiation of basic nuclear physics models, along with reactor parameters stated in the input description, are used to instantiate the corresponding aggregate effects models. An aggregate effects model is instantiated when the requirements for the reactants, material, and equipment defined in the aggregate effects model are satisfied. The result of this instantiation is a set of conceivable processes. The NPT behaviour generator then proceeds to test the validity of the preconditions and activity conditions for each instance. This requires that the initial values for at least a subset of parameters and variables have been specified. The result of this test is a set of active processes.

The state space and probabilistic space equations always follow directly from the set of active processes. These are the consequences of a number of influences defined in the aggregate effects models of the active processes. When the behaviour generator generates the state space and probabilistic space equations of active processes, it uses product specifications defined in the basic nuclear physics models of the processes. NPT also requires the nuclide's attributes data which are represented as an open hash table of the Chart of the Nuclides [17].

Once the state space and probabilistic space equations have been created, the NPT interpreter evaluates these equations, and produces a set of product nuclides, product particles, energy of product nuclides, and energy of product particles along with the consumption of the appropriate nuclides. Both product nuclides and product particles will then instantiate basic nuclear physics and aggregate effects models of some of the processes in the next cycle, see Fig. 5.3. The state space and probabilistic space equations may change from cycle time to cycle time depending upon how many types of nuclides are available in the system. A computer program called NPTsim, a total

of 7200 lines of source code, was written in C to perform these tasks. The functional model of NPTsim is shown in Fig. 5.4. Another C program called MakeHash was written (total of 870 lines of source) to create the binary representation of Chart of Nuclide Data.

5.2 NPT language

This section describes the NPT language. The NPT language is designed for representing nuclear physics processes to allow the writing of computer programs which simulate nuclear physics interactions. The NPT language is defined by the Extended Backus-Naur Form (EBNF) grammar which is given in Appendix III. This grammar is made up of rules that specify how to form sentences in the language. The structure of the language processor for NPT is shown in Fig. 5.5. The language processor consists of a lexical analyzer, parser and behaviour generator. The lexical analyzer accepts strings written in the language and identifies substrings that are elements of the language. The substrings, called tokens, are passed onto the parser, whose responsibility it is to build structures representing the statements in the language.

The parser controls the lexical analyzer. The parser asks the lexical analyzer for a particular type of token. The lexical analyzer looks at the next program token, determines its type, and passes the token to the parser.

Starting with the start symbol of the grammar, the parser uses the tokenized source program to fulfill the requirements of the grammar. The parser disassembles statements in the language and produces, for each statement, a series of tokens. These tokens define the function to be executed and parameters of the function.

Once the statement has been parsed, we know what to do with this statement. This information (function and parameters) is saved until the program is executed by the behaviour generator. The NPT language processor runs the program and produces the behaviour of nuclear physics processes which is specified in the program.

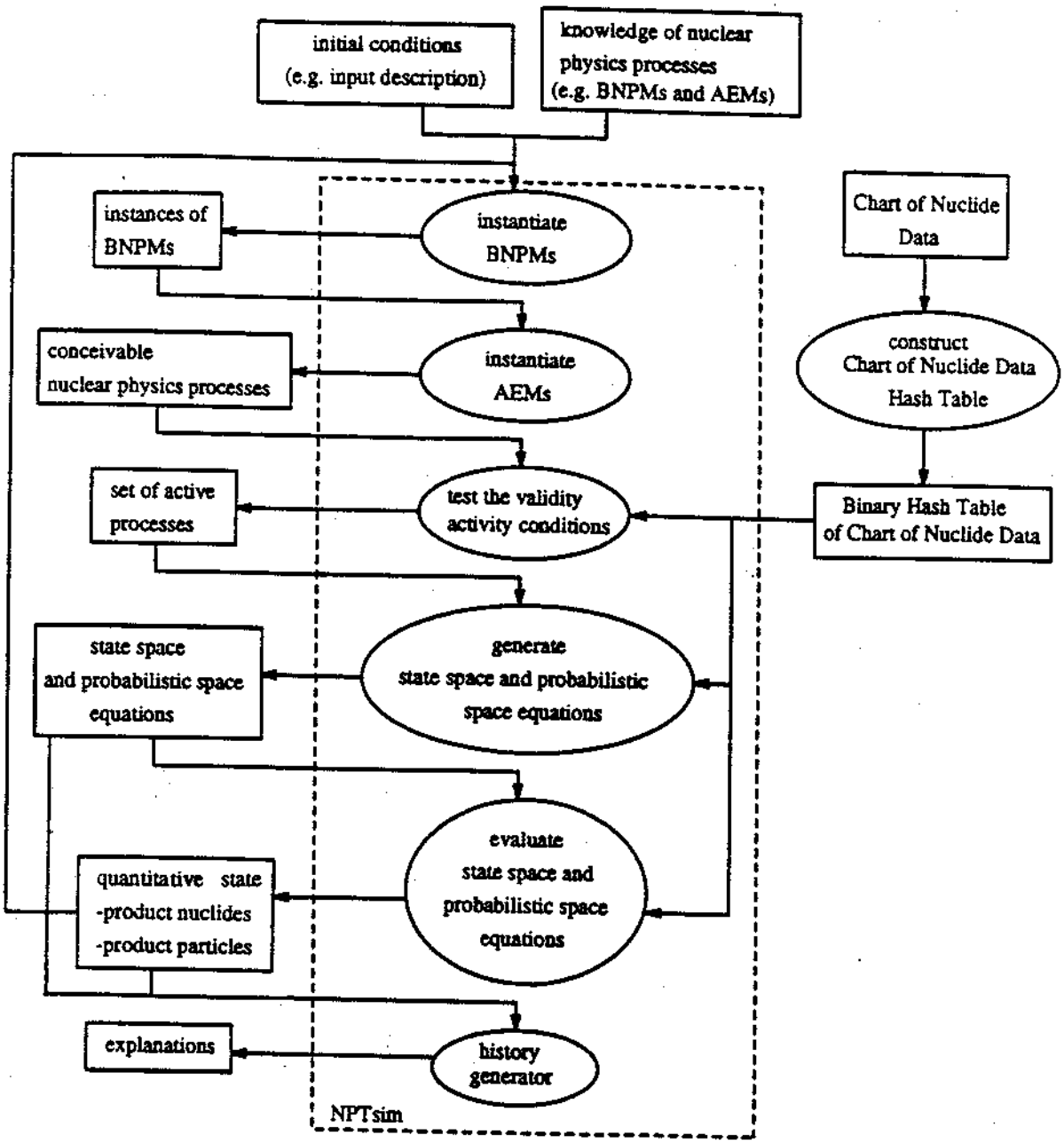

Figure 5.4: Functional model of NPTsim.

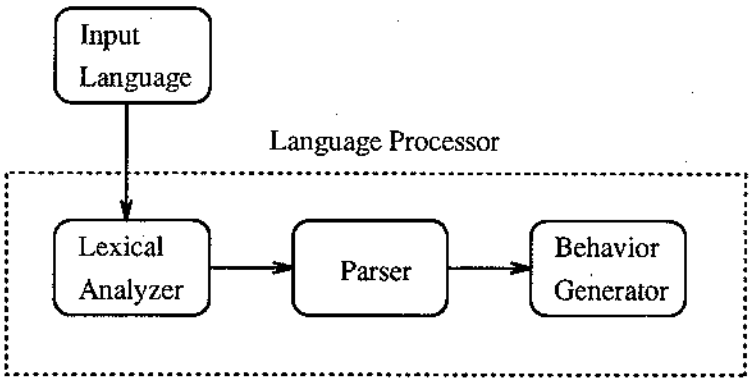

Figure 5.5: Basic organization of the NPT language processor.

The processing performed by the behaviour generator includes

1. For each type of nuclide, instantiate AEMs and BNPMs which produces a set of conceivable nuclear physics processes.

2. Determine the set of active processes by testing the validity of activity and preconditions.

3. Determine the type of the product nuclides of the processes being modelled using the product field in the BNPMs and the nuclide data [18].

4. Generate state space and probabilistic state space equations from all influences defined in AEMs, showing interactions between the processes being modelled.

5. Attach the type of active nuclide identity to the variable specified in the AEMs having symbolic names followed "(obj1)" or initialized by "obj1". Similarly, attach the actual name of product nuclides to the symbolic name of variables associated with the product nuclides. The actual name of a nuclide is obtained from the nuclide data [18].

6. Repeat the processes mentioned above for all existing types of nuclides at the current cycle. At the first cycle, the existing nuclides defined in the input description are used.

7. Once the processes mentioned above are completed, the interpreter evaluates any state space and probabilistic space equations at the current cycle. All storage memory is reset at the end of each cycle, except for the storage that deals with the reporting variables. This storage is reset at the end of each reporting cycle.

8. The above steps are repeated as long as the processes are active or for the duration of the simulation period.

Once the input description, (consisting of initial NPT objects, reactor, and cladding parameters) has been parsed, all these parameters are stored in memory. Once all basic nuclear physics and aggregate effects models have been parsed, the initial NPT objects stored in memory are checked against all reactants in the basic nuclear physics model. If they satisfy the reactants fields, they instantiate a set of BNPMs. To instantiate the AEMs, the requirements for materials and equipment stated in the AEMs must be satisfied. These will produce a set of conceivable nuclear processes.

In order to have a set of active processes, step 2 mentioned above is carried out. Having all influences defined in the AEMs, a set of probabilistic and state space equations is produced showing the interaction between reagents (e.g. how many nuclides interact with neutrons in both the fission process, and the radioactive capture process when one or more types of nuclides are active).

For example, assuming that we have four types of BNPM and AEM models in NPT, (i.e. fission, radioactive capture, beta positive and beta negative decays), neutrons can interact with nuclides through fission and radioactive capture only, since beta decay does not require neutrons. Assuming that two types of nuclides are available in the reactor core (e.g. ^{235}U and ^{238}U), the fraction of neutrons which interact with nuclides of type ^{238}U through radioactive capture will be generated as one of the state space equations as follows:

$$F_g U238 = U238.amount * U238.sigma\_g / Sigma\_a\_fuel.$$

The fraction of neutrons which interact with nuclides ^{235}U through the fission process will be generated as another state space equation, i.e.

$$F_f U235 = U235.amount * U235.sigma\_f / Sigma\_a\_fuel.$$

The generated state space equation for Sigma_a_fuel, at the cycle when only ^{235}U and ^{238}U are active, is given by

$$Sigma\_a\_fuel = U235.amount * U235.sigma\_f + U238.amount * U238.sigma\_g$$

The number of fissions at the current cycle is influenced by k, eta, the number of neutrons at the previous cycle neutron.amount, and the fraction of neutrons which interact with nuclides FfU235 (see Fig. 4.6). When^{235}U activates a fission process, the generated state space for the number of fissions is given by

$$RcU235 = k/eta * neutron.amount * FfU235$$

The neutrons produced by these fissions are influenced by the number of fissions and the probabilistic density of neutrons specified by pdf_neutron. The generated state space equations for distribution of neutron products, when ^{235}U activates fission is given by

$$RcU235 * pdf\_neutron[i], i = 0..8$$

The probabilistic space and state space equations are then evaluated by carrying out two tasks: a conversion from infix to postfix expression, and a postfix evaluation. These are due to the fact that each expression is stored in the node of a linked-list, and brackets are allowed in the expression. For example, the state space equation below

$$fU235 = U235.amount * (U235.sigma\_f + U235.sigma\_g)/ \\ (Sigma\_a\_fuel + Sigma\_a\_cl + Sigma\_a\_md)$$

is stored in the one of the nodes of the linked-list.

Chapter 6

Evaluation

This section illustrates how NPT works by showing three examples of the fission process and other related processes in the core of a reactor. In the first experiment, the reactor was fueled with pure ^{235}U. In the second experiment, natural Uranium, which consists of 0.7 % of $^{235}_{92}U$ and 99.3 % of $^{238}_{92}U$, is used as the fuel of the reactor. For the third experiment, the reactor was fueled by an enriched natural Uranium which consists of 12 % of $^{235}_{92}U$ and 88 % of $^{238}_{92}U$. The average prompt neutron life time lp (i.e. $20ms$) was adopted as the cycle time for the NPT simulator. In these experiments, NPT was used to predict, after a period of time, how much fuel is consumed, how much fuel is left, the composition of the product nuclides and the amount of energy produced.

Only the second experiment results are discussed in detail here. The two other experiments are discussed in [17]. There are many other processes accompanying the fission process [21]. For the experiments discussed here, only four processes are considered; they are fission, radioactive capture, beta negative decay and beta positive decay. Hence the nuclear physics system Q described in section 3.1 consists of four types of processes q_1, q_2, q_3 and q_4. The BNPMs and AEMs definition for these

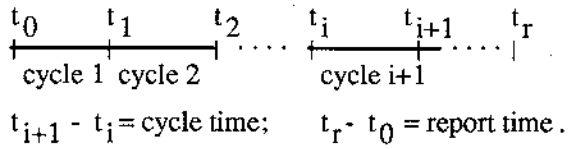

Figure 6.1: Time line used in NPT.

processes are shown in Fig. 4.2 through Fig. 4.10. The experiment was carried out over a period of 20 seconds. The time line used by NPT is shown in Fig. 6.1, where $t_i - t_{i-1}$ and $t_r - t_o$ represent the cycle time, and report time respectively. The reporting variables are reset every report time, while the cycle variables are reset every cycle time.

6.1 Experiments

In experiment 2, a reactor core is fueled by 1000 kg of natural Uranium which consists of 0.7 % of ${}_{92}U^{235}$ and 99.3 % of ${}_{92}U^{238}$. Other experiments are discussed in [17]. The input descriptions to the NPTsim program, including reactor parameters are shown in Appendix II. Part of the input values are taken from [21]. This section describes how NPTsim works given a mixture of substances as fuel.

Assuming that all relevant basic nuclear physics and aggregate effects models are included in the system, NPT identifies all possible instantiations of the basic nuclear physics and aggregate effects models in a manner similar to that discussed in Section 5.1.

In this experiment there are two types of nuclides, ^{235}U and ^{238}U, initially available in the reactor core. As can be seen in Fig. 3.1, only the nuclides of type ^{235}U can interact with neutrons through fission as indicated by the value of fission cross section. Both nuclide types ^{235}U and ^{238}U can interact with neutrons through the radioactive capture process as indicated by the values of radioactive capture cross sections.

Since only two types of nuclides are available initially, only two types of nuclides

activate the defined processes. Each type of nuclide activates BNPMs of the defined four processes, which in turn activate the corresponding AEMs.

The activity conditions at the current cycle are then tested to determine the active processes. The results are a set of active processes. The active processes at the beginning of first cycle are shown in Fig. 6.2.

```
U235
neutron
PHWNUR
HW
Zra
BNPM radioactive_capture(U235,neutron)
AEM (BNPM radioactive_capture(U235,neutron))
BNPM fission(U235,neutron)
AEM (BNPM fission(U235,neutron),PHWNUR,HW,Zra)
U238
BNPM radioactive_capture(U238,neutron)
AEM (BNPM radioactive_capture(U238,neutron))
```

Figure 6.2: Active processes at the beginning of the first cycle of the Experiment.

From these active processes, the behaviour generator produces a set of probabilistic and state space equations such as those shown in Fig. 6.3 and Fig. 6.4.

At t_1 (see Fig. 6.1), besides nuclides ^{235}U and ^{238}U, there are fission products from fissioning, and product nuclides from capturing neutrons by nuclides of types ^{235}U and ^{238}U through radioactive capture processes. Hence, not only ^{235}U and ^{238}U activate the basic nuclear physics and aggregate effects models of fission and radioactive capture, but also some fission products and product nuclides of radioactive capture processes. Each type of nuclide instantiates the appropriate basic nuclear physics model and the corresponding aggregate effects models. Therefore, there are many active processes at the end of the second cycle as given in Appendix IV, part of which are shown in Fig. 6.5.

At the end of the second cycle t_2, there are new nuclides born as products of

```
Sigma_a_fuel = U235.amount*U235.sigma_g+U235.amount*U235.sigma_f+
U238.amount*U238.sigma_g;

FgU235 = U235.amount*U235.sigma_g/Sigma_a_fuel;

nrcU235 = neutron.amount*k/eta*FgU235;

PrcU235 = nrcU235*U235.A/NA;

CrcU235 = nrcU235*U235.A/NA;

U235.amount += -1.00*CrcU235-CfU235;

U236.amount += 1.00*PrcU235;

Gamma.energy += 1.98*nrcU235+sum((i+ei/2)*distr_E_GammaU235[i],0,8)+
1.98*nrcU238;

Sigma_f_fuel = U235.amount*U235.sigma_f;

fU235 = U235.amount*(U235.sigma_f+U235.sigma_g)/
(Sigma_a_fuel+Sigma_a_cl+Sigma_a_md);

eta = nuU235*U235.amount*U235.sigma_f/Sigma_a_fuel;

k = fU235*eta*RC.epsilon*RC.p*RC.PNLf*RC.PNLth;

FfU235 = U235.amount*U235.sigma_f/Sigma_a_fuel;

RcU235 = k/eta*neutron.amount*FfU235;

CfU235 = RcU235/NA*U235.A;

distr_neutronU235[i] = RcU235*pdf_neutronU235[i],0,8;
```

Figure 6.3: State and probabilistic space equations at the end of the first cycle of Experiment 2.

```
pnU235 = sum(i*distr_neutronU235[i],0,8);

distr_A1U235[i] = RcU235*pdf_A1U235[i],0,44;

distr_A2U235[i] = RcU235*pdf_A2U235[i],0,44;

distr_E_neutronU235[i] = pnU235*pdf_E_neutronU235[i],0,8;

distr_E_gammaU235[i] = RcU235*pdf_E_gammaU235[i],0,8;

distr_E_F1U235[i] = RcU235*pdf_E_F1U235[i],0,44;

distr_E_F2U235[i] = RcU235*pdf_E_F2U235[i],0,44;

Fissile_fissioned += RcU235;

Fissile_consumed += CfU235;

Neutron.amount += pnU235;

Neutron.energy += sum((i+ei/2)*distr_E_neutronU235[i],0,8);

Energy_F1.amount += sum((i+ei/2)*distr_E_F1U235[i],0,90);

Energy_F2.amount += sum((i+ei/2)*distr_E_F2U235[i],0,90);

FgU238 = U238.amount*U238.sigma_g/Sigma_a_fuel;

nrcU238 = neutron.amount*k/eta*FgU238;

PrcU238 = nrcU238*U238.A/NA;

CrcU238 = nrcU238*U238.A/NA;

U238.amount += -1.00*CrcU238;

U239.amount += 1.00*PrcU238;
```

Figure 6.4: Continuation of state and probabilistic space equations at the end of the first cycle of Experiment 2.

```
U235
neutron
PHWNUR
HW
Zra
BNPM radioactive_capture(U235,neutron)
AEM (BNPM radioactive_capture(U235,neutron))
BNPM fission(U235,neutron)
AEM (BNPM fission(U235,neutron),PHWNUR,HW,Zra)
U238
BNPM radioactive_capture(U238,neutron)
AEM (BNPM radioactive_capture(U238,neutron))
U236
BNPM radioactive_capture(U236,neutron)
AEM (BNPM radioactive_capture(U236,neutron))
U239
BNPM beta_negative_decay(U239)
AEM (BNPM beta_negative_decay(U239))
Tb158
BNPM beta_negative_decay(Tb158)
....
```

Figure 6.5: Portion of active processes at the end of the second cycle of Experiment 2.

processes, which can activate different processes than occur in the previous cycle. The active processes usually change from cycle to cycle due to the fact that new nuclides may be born or existing nuclides may vanish. The vanished nuclides cause the corresponding active processes to be removed. The change of active processes leads to a dynamic set of state space and probabilistic space equations.

The active processes at the end of the second cycle cause the behaviour generator to produce different types of nuclides. Some of them have the same type as the available nuclides, some are new. If the product nuclides are not new, the number of nuclides are accumulated with the previous nuclides. If they are new, the number of nuclides are stored in new memory locations. Both the new and the previous nuclides

will activate some BNPMs and AEMs which lead to some active processes. These active processes produce different state space and probabilistic space equations from the previous cycle. The above activities are repeated over the period of simulation, or as long as the pre- and activity conditions are satisfied. The reporting variables, defined by the user, are reported every reporting period.

6.1.1 State space and probabilistic space equations of the first cycle for experiment 2

Since initially a mixture of ^{235}U and ^{238}U is available as a fuel in the reactor, the state space equation for Σ_a^{fuel}, appears in Fig. 6.3 as equation (6.1). This equation is the result of ^{235}U activating the basic nuclear physics model of fission and the corresponding aggregate effects model; and of ^{235}U and ^{238}U activating the basic nuclear physics models of radioactive capture, and the corresponding aggregate effects models.

$$Sigma\_a\_fuel = U235.amount * U235.sigma\_g + U235.amount * U235.sigma\_f + U238.amount * U238.sigma\_g \quad (6.1)$$

The first, the second and the third terms of the above equation show the results of interactions between neutrons and nuclides of type ^{235}U through the radioactive capture process, and the fission process, and the result of interaction between neutrons and nuclides of type ^{238}U through the radioactive capture process, respectively.

Since initially only nuclides of ^{235}U are available as fissionable nuclides, the state space equation Σ_f^{fuel}, as a result of ^{235}U activating the basic nuclear physics model and aggregate effects model of fission, is generated as

$$Sigma\_f\_fuel = U235.amount * U235.sigma\_f.$$

A fraction of neutrons that interacts with ^{235}U through radioactive capture is

generated as a state space equation

$$FgU235 = U235.amount * U235.sigma\_g/Sigma\_a\_fuel.$$

A state space equation

$$PrcU235 = FgU235 * neutron.amount * k/eta * U235.A/NA$$

is the product of neutrons captured by ^{235}U during cycle 1. This can be seen in Fig. 6.3 on lines 2 and 4. The consumption of ^{235}U during this cycle is generated as

$$CrcU235 = FgU235 * neutron.amount * k/eta * U235.A/NA$$

The radioactive capture process increases the mass number of the target nuclide by one, producing a new nuclide ^{236}U. The products of radioactive capture processes increase the amount of the resulting nuclides by an amount PrcU235, which appears as a state space equation

$$U236.amount += 1.00 * PrcU235. \tag{6.2}$$

The constant 1.00 in equation (6.2) indicates that one nuclide is produced for every radioactive capture process, as specified in the products section of the basic radioactive capture process.

The generated state space equation

$$U235.amount += -CrcU235 - CfU235, \tag{6.3}$$

shows the amount of ^{235}U consumed every cycle time period, both by the fission process (CfU235) and by the radioactive capture process (CrcU235), where the consumption of ^{235}U for fission, CfU235, is generated as a state space equation

$$CfU235 = RcU235/NA * U235.A,$$

where RcU235 is generated as in equation (6.4).

The "+=" operators in equations (6.3) and (6.2) are used to denote accumulation processes. These two equations show that any consumption will reduce the amount of the original nuclides, and any production will increase the amount of the resulting nuclides.

When only nuclides of ^{235}U are available as fissionable nuclides, then the thermal utilization factor f, the thermal fission factor η and the multiplication factor k are generated from state space equations as

$$fU235 = U235.amount{*}U235.sigma\_f/(Sigma\_a\_fuel{+}Sigma\_a\_md{+}Sigma\_a\_cl)\ ,$$

$$eta = nuU235 * U235.amount * U235.sigma\_f/Sigma\_a\_fuel\ ,$$

$$k = fU235 * eta + RC.epsilon + RC.p * RC.PNLf * RC.PNLth.$$

The fraction of neutrons which interact with ^{235}U through fission processes is generated as

$$FfU235 = U235.amount * U235.sigma\_f/Sigma\_a\_fuel.$$

This fraction of neutrons determines the number of fissions in the current cycle. When only nuclides ^{235}U and neutron.amount neutrons are available in the reactor core, then the number of fissions at the current cycle period is generated as a state space equation

$$RcU235 = k/eta * neutron.amount * FfU235. \tag{6.4}$$

These fissions influence the amount of ^{235}U consumed, which is generated in the state space equations as

$$CfU235 = RcU235/NA * U235.A.$$

The number of fissions RcU235 above determines the distribution of fission neutrons, the energy distribution of fission neutrons, the distribution of mass number of product nuclides, as well as the distribution of gamma energy. Since the probability of producing a certain number of neutrons and the probability of producing gamma energy

in a fission are represented as pdf_neutron and pdf_E_gamma respectively, the probabilistic space equations for the distribution of fission products and for the distribution of gamma energy are generated as

$$distr\_E\_gammaU235[i] = RcU235 * pdf\_E\_gammaU235[i], 0..8$$

and

$$distr\_neutronU235[i] = RcU235 * pdf\_neutronU235[i], 0..8,$$

respectively.

The number of fission neutrons as a result of fissioning ^{235}U is generated in the probabilistic space model as

$$pnU235 = sum(i * distr\_neutronU235[i]), 0..8. \tag{6.5}$$

The total number of fission neutrons resulting from fissioning different types of nuclides is accumulated in neutron.amount for each cycle and in Neutron.amount for each time report period. The number of neutrons, which comes from interactions between neutrons and ^{235}U increases both neutron.amount and Neutron.amount and is generated as

$$neutron.amount + = pnU235$$

and

$$Neutron.amount + = pnU235,$$

respectively. The difference between the two lies in the fact that neutron.amount is used to store the number of neutrons in the reactor core for one cycle time period whereas Neutron.amount is used to store neutrons generated within one report time period.

Each fission neutron is emitted with a continuous distribution of energy. The probability of a fission neutron having an energy in a certain interval is represented by pdf_E_neutron. The number of fission neutrons influences the energy distribution

of fission neutrons. The energy distribution resulting from fissioning ^{235}U is generated as

$$distr\_E\_neutronU235[i] = pnU235 * pdf\_E\_neutronU235[i], 0..8.$$

In NPT, each type of nuclide which interacts with neutrons through fission, is modeled to produce the associated fission neutrons along with its energy per cycle time. This energy is spread over certain intervals and is represented by distr_E_neutrons. The amount of energy of fission neutrons per cycle, as a result of fissioning ^{235}U, is accumulated over all energy ranges in distr_E_neutrons as follows:

$$sum((i + ei/2) * distr\_E\_neutronsU235[i]), 0..8$$

where ei is the neutron energy interval appearing in pdf_E_neutrons. Since the total amount of the energy of fission neutrons can come as a result of fissioning different types of nuclides, then the probabilistic state space equation for the total amount of the energy of fission neutrons (from ^{235}U) per cycle is generated as

$$Neutron.energy + = \ sum((i + ei/2) * distr\_E\_neutronsU235[i]), 0..8.$$

The number of fissions per cycle period, RcU235, influences the distribution of mass numbers of fission products. Since the probability of having a certain mass number is represented by vectors pdf_A1 and pdf_A2 for fission products F1 and F2, respectively, the distribution of mass numbers of fission products F1 and F2, caused by fissioning ^{235}U only, are generated in the probabilistic space model as

$$distr\_A1U235[i] = RcU235 * pdf\_A1U235[i], 73..117 \tag{6.6}$$

$$distr\_A2U235[i] = RcU235 * pdf\_A2U235[i], 118..162. \tag{6.7}$$

In order to identify a particular nuclide, its mass number A and its atomic number Z must be known. From the distribution of mass number of fission fragments obtained, a set of mass numbers $A_1, A_2 \cdots A_n$ are known. For each A_i , there is a set of Zs

associated with it. The distribution for Z is approximated by *proton-neutron* ratio [25], which is defined as follows:

$$Z_i = (Z_{cn}/(A_{cn} - \nu)) * A_i + \delta_Z,\ i = 73..162, \tag{6.8}$$

where Z_{cn} and A_{cn} are the atomic number and the mass number of the compound nucleus formed from a target nuclide and a neutron. In the above equation, δ_Z is non-deterministic. Its value is controlled by the Gaussian distribution. The term δ_Z represents the deviation of the product nuclide (Z_i, A_i) from the stable zone of nuclides, and ν is the average product neutrons per fission. The probability of δ_Z having a certain value is computed by decomposing the area under the Gaussian distribution curve. This probability distribution is represented as vector pdf_dZ

Once the number of fission fragments (product nuclei) having a particular mass number is known, then their isobars, (i.e. nuclides which have the same mass numbers A, but different atomic numbers) can be obtained easily using equation (6.8). The value for δ_Z for the jth isobar in the isobar line (see Fig. 6.6) is given by the index j of pdf_dZ. The number of nuclides for the jth isobar is influenced by the number of nuclides having mass number A and the element pdf_dZ[j].

The number of nuclides having a certain mass number A, as a result of the fission process, is represented as the element of distr_A. A itself is represented by the index of distr_A. Knowing the pdf_dZ, the number of nuclides of the jth isobar (Z_j, A) having mass number A and atomic number Z_j is computed using

$$distr\_A[A] * pdf\_dZ[j]$$

The value of Z_j is computed using equation (6.8). The value of δ_Z in equation (6.8) is equal to j.

As an illustration, consider a portion of the distribution of mass numbers, distr_A, of fission products tabularized in Table 6.1. Consider also the part of elements of pdf_dZ tabularized in Table 6.2. The number of nuclides having A=75 and located at

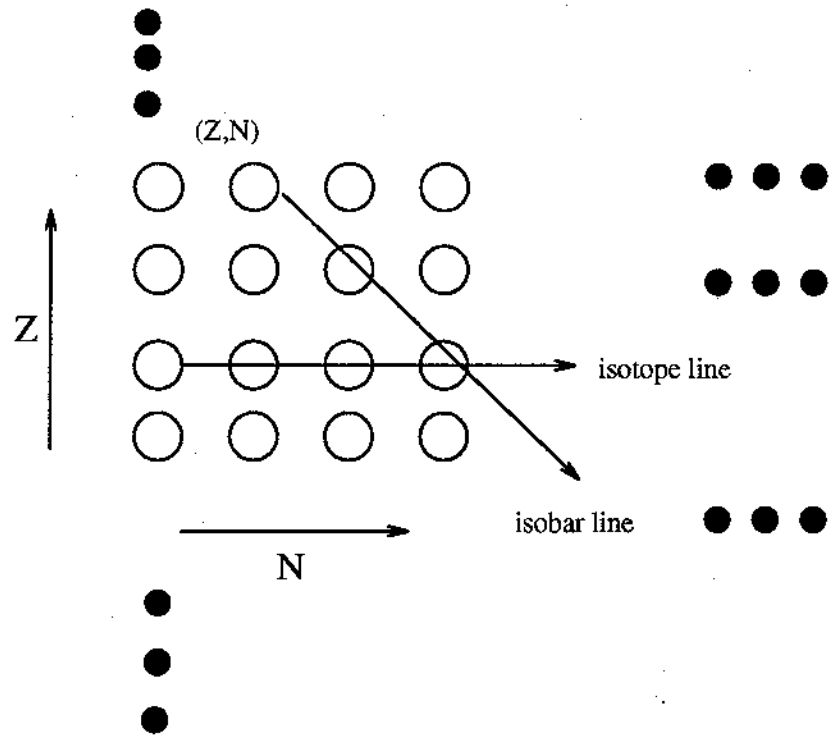

Figure 6.6: Isobar line.

the $j = -2$th element of the isobar line is equal to

$$distr\_A[75] * pdf\_dZ[-2] = 1000 * 0.0219 \tag{6.9}$$

with δ_Z equal to -2. The associated atomic number Z is computed using equation (6.8). The associated neutron number is computed as Z - A.

Knowing the distribution of mass numbers of product nuclides which are represented as distr_A1 and distr_A2 for fission products F1 and F2, respectively, then all isobars associated with each mass number can be generated based on equation (6.8). The set of distribution representations, i.e., pdf_dZ , distr_A1, distr_A2 as well as the knowledge about a compound nucleus and the average number of fission neutrons influence the generation of isobars of fission products through **distr_infl** . Each mass number corresponds to set of isobars.

For the fissioning ^{235}U, the NPT behaviour generator generates probabilistic space equations for all isobars of fission products. For program efficiency consideration, the

Table 6.1: Portion of distr_A.

index	number of nuclides
73	400
74	800
75	1000
.	
.	
.	

Table 6.2: Portion of pdf_dZ.

j	*probability* of δ_Z
-4	0.00001
-3	0.00104
-2	0.02190
0	0.14007
.	0.67902
.	0.13634

isobars are sorted in an ascending order based on the nuclide name. The probabilistic space equations for the sorted isobars are shown in Appendix VI, and part of them are shown in Fig. 6.7.

The number of fissions per cycle period, RcU235 discussed above, influences the distribution of energy of fission products. The probability of having a certain energy is represented by vectors pdf_E_F1 and pdf_E_F2 for fission products F1 and F2, respectively. The distribution of energy of fission products F1 and F2 appears in the probabilistic space equations as

$$distr\_E\_F1U235 = RcU235 * pdf\_E\_F1U235[i], 73..117,$$

$$distr\_E\_F2U235 = RcU235 * pdf\_E\_F2U235[i], 118..162.$$

The number of fissions of fissioning many types of nuclides, $Rc_j, (j = 1..n)$, per

```
Ag109.amount += distr_A1U235[110]*pdf_dZ[4]/NA*110;
Ag110.amount += distr_A1U235[111]*pdf_dZ[4]/NA*111;
Ag111.amount += distr_A1U235[112]*pdf_dZ[3]/NA*112;
Ag112.amount += distr_A1U235[113]*pdf_dZ[3]/NA*113;
Ag113.amount += distr_A1U235[114]*pdf_dZ[3]/NA*114;
Ag114.amount += distr_A1U235[115]*pdf_dZ[2]/NA*115;
Ag115.amount += distr_A1U235[116]*pdf_dZ[2]/NA*116;
Ag116.amount += distr_A1U235[117]*pdf_dZ[1]/NA*117;
Ag117.amount += distr_A2U235[118]*pdf_dZ[1]/NA*118;
Ag118.amount += distr_A2U235[119]*pdf_dZ[1]/NA*119;
Ag119.amount += distr_A2U235[120]*pdf_dZ[0]/NA*120;
Ag120.amount += distr_A2U235[121]*pdf_dZ[0]/NA*121;
Ag121.amount += distr_A2U235[122]*pdf_dZ[-1]/NA*122;
Ag122.amount += distr_A2U235[123]*pdf_dZ[-1]/NA*123;
Ag123.amount += distr_A2U235[124]*pdf_dZ[-1]/NA*124;
Ag124.amount += distr_A2U235[125]*pdf_dZ[-2]/NA*125;
Ag125.amount += distr_A2U235[126]*pdf_dZ[-2]/NA*126;
Ag126.amount += distr_A2U235[127]*pdf_dZ[-3]/NA*127;
Ag127.amount += distr_A2U235[128]*pdf_dZ[-3]/NA*128;
Ag128.amount += distr_A2U235[129]*pdf_dZ[-3]/NA*129;
Ag129.amount += distr_A2U235[130]*pdf_dZ[-4]/NA*130;
Ag130.amount += distr_A2U235[131]*pdf_dZ[-4]/NA*131;
Ag131.amount += distr_A2U235[132]*pdf_dZ[-4]/NA*132;
As73.amount += distr_A1U235[74]*pdf_dZ[4]/NA*74;
As74.amount += distr_A1U235[75]*pdf_dZ[4]/NA*75;
As75.amount += distr_A1U235[76]*pdf_dZ[4]/NA*76;
As76.amount += distr_A1U235[77]*pdf_dZ[3]/NA*77;
As77.amount += distr_A1U235[78]*pdf_dZ[3]/NA*78;
As78.amount += distr_A1U235[79]*pdf_dZ[2]/NA*79;
As79.amount += distr_A1U235[80]*pdf_dZ[2]/NA*80;
As80.amount += distr_A1U235[81]*pdf_dZ[2]/NA*81;
As81.amount += distr_A1U235[82]*pdf_dZ[1]/NA*82;
As82.amount += distr_A1U235[83]*pdf_dZ[1]/NA*83;
As83.amount += distr_A1U235[84]*pdf_dZ[0]/NA*84;
As84.amount += distr_A1U235[85]*pdf_dZ[0]/NA*85;
As85.amount += distr_A1U235[86]*pdf_dZ[0]/NA*86;
As86.amount += distr_A1U235[87]*pdf_dZ[-1]/NA*87;
As87.amount += distr_A1U235[88]*pdf_dZ[-1]/NA*88;
As88.amount += distr_A1U235[89]*pdf_dZ[-2]/NA*89;
As89.amount += distr_A1U235[90]*pdf_dZ[-2]/NA*90;
```

Figure 6.7: A portion of probabilistic space equations for fission products at t_1 for experiment no. 2.

```
As90.amount += distr_A1U235[91]*pdf_dZ[-2]/NA*91;
As91.amount += distr_A1U235[92]*pdf_dZ[-3]/NA*92;
As92.amount += distr_A1U235[93]*pdf_dZ[-3]/NA*93;
As93.amount += distr_A1U235[94]*pdf_dZ[-4]/NA*94;
As94.amount += distr_A1U235[95]*pdf_dZ[-4]/NA*95;
As95.amount += distr_A1U235[96]*pdf_dZ[-4]/NA*96;
Ba132.amount += distr_A2U235[133]*pdf_dZ[4]/NA*133;
Ba133.amount += distr_A2U235[134]*pdf_dZ[4]/NA*134;
Ba134.amount += distr_A2U235[135]*pdf_dZ[3]/NA*135;
Ba135.amount += distr_A2U235[136]*pdf_dZ[3]/NA*136;
Ba136.amount += distr_A2U235[137]*pdf_dZ[3]/NA*137;
Ba137.amount += distr_A2U235[138]*pdf_dZ[2]/NA*138;
Ba138.amount += distr_A2U235[139]*pdf_dZ[2]/NA*139;
Ba139.amount += distr_A2U235[140]*pdf_dZ[1]/NA*140;
Ba140.amount += distr_A2U235[141]*pdf_dZ[1]/NA*141;
Ba141.amount += distr_A2U235[142]*pdf_dZ[1]/NA*142;
Ba142.amount += distr_A2U235[143]*pdf_dZ[0]/NA*143;
Ba143.amount += distr_A2U235[144]*pdf_dZ[0]/NA*144;
Ba144.amount += distr_A2U235[145]*pdf_dZ[-1]/NA*145;
Ba145.amount += distr_A2U235[146]*pdf_dZ[-1]/NA*146;
.....
```

Figure 6.8: Continuation of a portion of probabilistic space equations for fission products at t_1 for experiment no. 2.

report time period are accumulated in Fissile_fissioned. Since only nuclides of ^{235}U activate the fission process in the first cycle, then the state space equation for Fissile_fissioned is generated as

$$Fissile\_fissioned += RcU235.$$

A state space equation

$$Fissile\_consumed += -CfU235.$$

is generated showing the amount of substance consumed per cycle period $Cf_{j=1..n}$ which are accumulated in Fissile_consumed.

The nuclides of types ^{238}U and ^{235}U activate radioactive capture processes. When these processes are active, the nuclides of type ^{235}U cause the NPT interpreter to generate state space equations for a fraction of neutrons for radioactive capture

FgU235, consumption of ^{235}U CrcU235, productions of ^{236}U PrcU235 which are caused by nuclides of ^{235}U capturing neutrons, the amount of neutron capturing nuclides U235.amount, and the number of the resulting nuclides U236.amount. Similarly, the nuclides of type ^{238}U cause the NPT interpreter to generate state space equations for FgU238, PrcU238, CrcU238, U238.amount, and U239.amount. The state space equations related to ^{238}U are shown below:

$$FgU238 = U238.amount * U238.sigma\_g/Sigma\_a\_fuel \tag{6.10}$$

$$PrcU238 = FgU238 * k/eta * neutron.amount \tag{6.11}$$

$$CrcU238 = FgU238 * k/eta * neutron.amount \tag{6.12}$$

$$U239.amount += 1.00 * PrcU238 \tag{6.13}$$

$$U238.amount += -1.00 * CrcU238 \tag{6.14}$$

The nuclides ^{238}U interact with neutrons through radioactive capture only; i.e. ^{238}U is consumed only for radioactive capture. The state space equation for ^{238}U consumption is different than that of ^{235}U consumption as seen by comparing equations (6.14) and (6.3).

6.1.2 State space and probabilistic space equations of the second cycle for experiment 2

At t_1, there are many types of product nuclides other than ^{235}U. The product nuclides ^{236}U come from radioactive capture involving neutrons from nuclides of ^{235}U. The fission products with mass numbers ranging from 73 to 162 come from the result of fissioning ^{235}U. The product nuclides of ^{239}U come from the radioactive capture of neutrons with nuclides of ^{238}U. It can be seen from Fig. 3.1 that the nuclide ^{239}U can interact with neutrons through fission, through radioactive capture, and that it undergoes beta negative decay. Consequently, ^{239}U instantiates three types of basic

nuclear and aggregate effect models: fission, radioactive capture and beta negative decay. Both ^{235}U and ^{239}U interact with neutrons through the fission process. Nuclides ^{235}U, ^{236}U and ^{239}U interact with neutrons through radioactive capture, along with other types of nuclides which come from fissioning ^{235}U at the first cycle; i.e. Tb159, Gd156, Gd157, Eu153, Eu154, Eu155, Sm150, Sm151, Sm152, Sm153, Sm154, Pm148, Nd145, Nd146, Nd148, Nd150, Pr143, La138, Cs133, Xe130, Xe131, Xe135, I127, I129, Sb123, Mo100, and Kr85. The result of the above nuclides interacting with neutrons through radioactive capture, and both nuclides ^{239}U and ^{235}U interacting through the fission process can be seen in the generated state space equation of Σ_a^{fuel} shown in Fig. 6.9. The resulting state space and probabilistic space equations at time t_2 of Experiment 2 are given in Appendix V, while a portion of them is shown in Fig. 6.9. The complete set of active processes in the second cycle can be seen in Appendix IV.

The nuclide ^{239}U and some product nuclides, which come from fissioning ^{235}U at the first cycle, are not stable. They undergo either beta negative decay or beta positive decay by emitting beta particles along with beta energy. The corresponding actions in NPT are that each type of nuclide activates the basic nuclear physics model of beta negative decay or beta positive decay, and the associated aggregate effects model. The state space equations of nuclide production and consumption as a result of each type of nuclide is produced. The number of beta particles produced, and the energy of beta released by each type of nuclide is accumulated in the state space equations Beta_Neg.amount and Beta_Neg.energy (see Appendix V).

When a simulation is carried out, the resulting state space equations change from cycle to cycle depending upon how many types of nuclides are available in the reactor core, and how many types of interactions occur. The results of running a simulation for 20 seconds are shown in Appendix VII.

The results of the evaluations of the state space and the probabilistic equations are reported every report time period. The reported result includes the number of

```
Sigma_a_fuel = U235.amount*U235.sigma_g+U235.amount*U235.sigma_f+
U236.amount*U236.sigma_g+U238.amount*U238.sigma_g+Tb159.amount*
Tb159.sigma_g+Gd155.amount*Gd155.sigma_g+Gd156.amount*Gd156.sigma_g+
Gd157.amount*Gd157.sigma_g+Eu153.amount*Eu153.sigma_g+Eu154.amount*
Eu154.sigma_g+Eu155.amount*Eu155.sigma_g+Sm150.amount*Sm150.sigma_g+
Sm151.amount*Sm151.sigma_g+Sm152.amount*Sm152.sigma_g+Sm153.amount*
Sm153.sigma_g+Sm154.amount*Sm154.sigma_g+Pm148.amount*Pm148.sigma_g+
Nd145.amount*Nd145.sigma_g+Nd146.amount*Nd146.sigma_g+Nd148.amount*
Nd148.sigma_g+Nd150.amount*Nd150.sigma_g+Pr143.amount*Pr143.sigma_g+
La138.amount*La138.sigma_g+Cs133.amount*Cs133.sigma_g+Xe130.amount*
Xe130.sigma_g+Xe131.amount*Xe131.sigma_g+Xe135.amount*Xe135.sigma_g+
I127.amount*I127.sigma_g+I129.amount*I129.sigma_g+Sb123.amount*
Sb123.sigma_g+Mo100.amount*Mo100.sigma_g+Kr85.amount*Kr85.sigma_g;

FgU235 = U235.amount*U235.sigma_g/Sigma_a_fuel;

nrcU235 = neutron.amount*k/eta*FgU235;

PrcU235 = nrcU235*U235.A/NA;

CrcU235 = nrcU235*U235.A/NA;

U235.amount += -1.00*CrcU235-CfU235;

U236.amount += 1.00*PrcU235-1.00*CrcU236;

Gamma.energy += 1.98*nrcU235+sum((i+ei/2)*distr_E_GammaU235[i],0,8)+
1.98*nrcU236+1.98*nrcU238+1.98*nrcTb159+1.98*nrcGd155+1.98*nrcGd156+
1.98*nrcGd157+1.98*nrcEu153+1.98*nrcEu154+1.98*nrcEu155+1.98*nrcSm150+
1.98*nrcSm151+1.98*nrcSm152+1.98*nrcSm153+1.98*nrcSm154+1.98*nrcPm148+
1.98*nrcNd145+1.98*nrcNd146+1.98*nrcNd148+1.98*nrcNd150+1.98*nrcPr143+
1.98*nrcLa138+1.98*nrcCs133+1.98*nrcXe130+1.98*nrcXe131+1.98*nrcXe135+
1.98*nrcI127+1.98*nrcI129+1.98*nrcSb123+1.98*nrcMo100+1.98*nrcKr85;

Sigma_f_fuel = U235.amount*U235.sigma_f;

fU235 = U235.amount*(U235.sigma_f+U235.sigma_g)/
(Sigma_a_fuel+Sigma_a_cl+Sigma_a_md);
...
```

Figure 6.9: Portion of the state and probabilistic space equations at time t_2 Experiment 2.

fissions over the report time period, amount of fissile material consumed for both ^{235}U and ^{238}U in units of *grams*, the number of fission neutrons along with the energy of fission neutrons, the energy of gamma products, the energy of fission fragments for both F1 and F2, the number of beta particles and the energy of beta particles. The distribution of nuclides in the reactor core are reported at the end of the simulation. These results show a subcritical state for the reactor, where the number of fissions decreases with each report period. The reports are shown up to the third report period only. The entire 20 second simulation required 2 hours, 4 minutes, and 1 second CPU time on SUN SPARC station IPX computer. This experiment required about half of the time required for the first and the third experiments [17]. This is due to the fact that after 9 seconds, no more fission processes occur, and the only remaining processes requiring calculations are the beta positive and beta negative decays.

6.2 Comparison to Expected Results

A theory is usually tested by comparing the observed actual behaviour of a system, sometimes resulting from a carefully constructed experiment, with the predictions made from the theory.

NPT can predict the behaviour of nuclear physics processes by generating dynamic state space equations and probabilistic equations every cycle and evaluating these equations. The results generated by NPT may be verified against experiments, but any discrepancy between the observed behaviour and the behaviour predicted by NPT does not necessarily indicate a flaw in the approach taken by NPT. Often, such discrepancies may be better explained by inadequate knowledge of nuclear physics processes (e.g., distribution of prompt gamma particles, distribution of product nuclides) or by inappropriate assumptions made by the modeler interacting with NPT (e.g. parameters of the reactor).

The examples shown in section 6.1 and in [17] demonstrate some of the capabilities of NPT and NPTsim. The example does exercise most of the NPT mechanisms and thus shows that the mechanisms work for nuclear physics systems which consist of nuclear physics processes (e.g., fission, radioactive capture, beta positive decay and beta negative decay).

For the experiment discussed in section 6.1, four basic nuclear physics and aggregate effects models were written. They are BNPMs and AEMs for fission, radioactive capture, β^+ decay and β^- decay. In this simulation, the reactor is fueled with 1000 kg of natural Uranium which consists of 0.7% of ${}_{92}U^{235}$ and 99.3% of ${}_{92}U^{238}$. The reactor parameters used for the simulation are 1.03 (fast fission factor), 0.985 (fast non-leakage probability), 0.667 (resonance escape probability) and 0.975 (thermal non-leakage probability). A geometric model has not yet been included in NPT. The reactor is assumed to be homogeneous, where there is no difference in flux due the distribution of fuel, moderator and cladding.

The experimental results for the $20th$ second are shown in Appendix VII. The report tells that 2.876e+05 nuclides were fissioned in the beginning, the number of nuclides fissioned decreases and they vanished after 9 seconds. These results showed a subcritical state for the reactor, where the number of fissions decreases with each report period. This is due the fact that the exact values for the resonance escape probability η and the resonance escape probability p are not known yet. The η and p adopted are the same values as for a homogeneous reactor, which are 1.03 and 0.667, respectively. For a heterogeneous reactor, such as the CANDU reactor, the value of η and p must be greater than that of a homogeneous reactor in order to achieve criticality [13]. The exact values are not yet reported. Experiment 3 reported in [17] shows that by using 12% enriched Natural Uranium as a fuel, the reactor remains critical for 20 seconds.

The energy released per fission in the beginning is 175.7 MeV with less contribution coming from the energy of beta particles. When the number of nuclides fissioned

decreases, more nuclides are involved in other processes such as the beta decay and the radioactive capture processes. These processes produce more energy than fission processes. This energy is the major contribution in the total energy released. The amount of energy released per fission in the form of kinetic energy of product nuclides (energy of F1 and F2) is found to be 167.02 MeV, i.e., within 0.02% error with respect to the experimental results presented in [21]. The amount of the average neutron energy released per fission is found to be 4.79 MeV, which means 4.14% error with respect to the experimental value of 5 MeV presented in [21]. This is partly due to the fact that a model for the delayed neutrons accompanying fission reactions has not been included in the system. The number of neutrons released per fission, ν, is found to be 2.429 which is very close to the known experimental value of 2.43 [21]. The value of ν will even be closer to 2.43 if a BNPM and AEM for the delayed neutron process is included. The available nuclides at the end of the simulation period are shown in Appendix VII.

6.3 Queries, Histories and Changes parameters

Having a set of both state space and probabilistic space equations, one can query, at a current cycle, either the value of a variable or the history of a variable to determine where it came from. To get the value of a variable, the NPT language allows one to write "*get_value* x", where x is the name of the variable being searched. For example, in order to know the amount of a particular type of nuclide (e.g. Krypton), the query "*get_value* $Kr85.amount$" will determine it.

To know how parameters of NPT change from cycle to cycle, the notion of history is used. History of a parameter is made up of state space and probabilistic space equations showing where it came from. The equations are stored in linked-lists, and when they are evaluated by the NPT behaviour generator, they require conversion from infix to postfix expressions. NPT takes into account a set of postfix expressions

```
History of Sigma_a_fuel
Sigma_a_fuel: U235.amount U235.sigma_f * U235.amount U235.sigma_g * +
U238.amount U238.sigma_g * +
U235.amount: (value: 1.200e+05)  CfU235 - CrcU235 -
CfU235: RcU235 NA / U235.A *
RcU235: k eta / Neutron.amount * FfU235 *
k: fU235 eta * HWRC.epsilon * HWRC.p * HWRC.PNLf * HWRC.PNLth *
fU235: U235.amount U235.sigma_f U235.sigma_g + * Sigma_a_fuel
Zacl.sigma_a + HWmd.sigma_a + /
U235.sigma_f: 5.850e+02 (leaf)
U235.sigma_g: 9.900e+01 (leaf)
Zacl.sigma_a: 0.185
HWmd.sigma_a: 0.001
eta: nu U235.amount * U235.sigma_f * Sigma_a_fuel /
nu: 2.43
HWRC.epsilon: 1.03
HWRC.p: 0.667
HWRC.PNLf: 0.865
HWRC.PNLth: 0.833
Neutron.amount: (value: 1.386e+20)  pnU235 + nrcU235 - nrcU238 -
pnU235: 0 distr_neutronU235[0] * 1 distr_neutronU235[1] * + 2
distr_neutronU235[2] * + 3 distr_neutronU235[3] * +
4 distr_neutronU235[4] * + 5 distr_neutronU235[5] * +
6 distr_neutronU235[6] * + 7 distr_neutronU235[7] * +
 8 distr_neutronU235[8] * +
distr_neutronU235[0]: RcU235 pdf_neutron[0] *
pdf_neutron[0]: 8.804e-02 (leaf)
distr_neutronU235[1]: RcU235 pdf_neutron[1] *
pdf_neutron[1]: 2.139e-01 (leaf)
distr_neutronU235[2]: RcU235 pdf_neutron[2] *
pdf_neutron[2]: 2.599e-01 (leaf)
distr_neutronU235[3]: RcU235 pdf_neutron[3] *
pdf_neutron[3]: 2.105e-01 (leaf)
distr_neutronU235[4]: RcU235 pdf_neutron[4] *
pdf_neutron[4]: 1.279e-01 (leaf)
distr_neutronU235[5]: RcU235 pdf_neutron[5] *
pdf_neutron[5]: 6.216e-02 (leaf)
distr_neutronU235[6]: RcU235 pdf_neutron[6] *
pdf_neutron[6]: 2.517e-02 (leaf)
distr_neutronU235[7]: RcU235 pdf_neutron[7] *
pdf_neutron[7]: 8.739e-03 (leaf)
```

Figure 6.10: Example of history and query.

```
distr_neutronU235[8]: RcU235 pdf_neutron[8] *
pdf_neutron[8]: 3.755e-03 (leaf)
nrcU235: Neutron.amount k * eta / FgU235 *
FgU235: U235.amount U235.sigma_g * Sigma_a_fuel /
nrcU238: Neutron.amount k * eta / FgU238 *
FgU238: U238.amount U238.sigma_g * Sigma_a_fuel /
U238.amount: (value: 8.800e+05)  CrcU238 -
CrcU238: nrcU238 U238.A * NA /
U238.A: 2.380e+02 (leaf)
NA: 6.02217e23
U238.sigma_g: 2.680e+00 (leaf)
FfU235: U235.amount U235.sigma_f * Sigma_a_fuel /
U235.A: 2.350e+02 (leaf)
CrcU235: nrcU235 U235.A * NA /

Get value: Sigma_a_fuel
Value: 8.44384e+07

History of Ag108.amount
Ag108.amount: 0.000e+00 (leaf)

Get value: U235.amount
Value: 1.20000e+05
```

Figure 6.11: Continuation of an example of history and query.

to create a history of a parameter. The history of a parameter at the current cycle is created by giving a keyword "history". For example, giving "*historySigma_a_fuel*" at the first cycle produces a history for parameter *Sigma_a_fuel*. This history is shown in Fig. 6.10. This history tells us that *Sigma_a_fuel* initially comes from $U235.amount$ * $U235.sigma\_f$ + $U235.amount$ * $U235.sigma\_g$ + $U238.amount$ * $U238.sigma\_g$, where $U235.amount$ comes from $-Cf235$ - $CrcU235$ and has a value of $U235.amout = 1.2e + 5$ as shown in Fig. 6.10. $cfU235$ comes from $RcU235 * NA/235.A$. Other sequences are shown in the Figures 6.10 and 6.11.

A value of a parameter of NPT can be changed directly by giving a keyword "*set_value*". For example, to change the parameter of $HWRC.p$ from the previous

value to the value of 0.98, one must type "*set_value HWRC.p* 0.98".

6.4 Worst Case Time Analysis

The running time of NPTsim can be analyzed in terms of the number of operations required to evaluate state space and probabilistic space equations in one cycle time. The analysis includes

1. Time required for instantiation of the BNPM and AEM for establishing active processes, which, in turn, are used to obtain the products of processes.

2. Time required for generating state space and probabilistic space equations.

3. Time required for evaluating state space and probabilistic space equations.

Item 1 requires $O(m)$ time for establishing active processes, and O(1) time for determining the products of processes (e.g. product nuclides). The term m corresponds to the number of available processes expressed in BNPM and AEM (in our example $m = 4$). $O(1)$ is the time required to determine a particular nuclide (e.g. ^{236}U) along with its features from the representation of nuclide data [18].

Generating a state space and probabilistic space equation requires $l * r * s$ comparisons, where l, r and s represent the number of characters in the equation, the length of a symbolic name of active objects (e.g. obj1, obj2), and the number of types of active nuclides. Therefore, the time to generate t equations is $O(t * l * r * s)$. For our example in the worst case, l can be as large as 100, r can be up to 5, s can be up to 3000 (all types of nuclides in the nuclide data), and t can be up to 10000 for all processes.

To evaluate the state space and probabilistic space equations requires two types of operations: a conversion from infix to postfix expression, and a postfix evaluation. Each operation requires a constant time $O(u)$ [32], where u is the number of terms

in the expression. The running time of the simulation using the NPT model for each cycle time is thus

$$O(m) + O(1) + O(t * l * r * s) * 2 * O(u),$$

where u is 10 in the worst case. This reduces to

$$O(t * l * r * s * u)$$

Substituting the above worst case values gives an order of 1.5×10^{11} operations per cycle time.

From the generated state space and probabilistic space equations, one can query either the value of a variable, or the history of a variable to determine where it came from. The worst case time to search for a value of a parameter is $O(t)$, because, in the worst case, the complete linked-list representing all influenced variables in the state space and probabilistic space equations must be traversed.

6.5 Generalization

NPT can be adapted for use with other processes outside the domain of nuclear physics.

6.5.1 Environmental Modeling

For example, environmental modeling deals with the simulation of environmental phenomena. From the perspective of a modeller, air pollution problems are conveniently characterized by their scale and by the types of pollutants involved. Many sources of pollutants are reported in [40], such as formaldehyde, NOx, carbon monoxide, and radon. The processes of producing these different types of pollutant can be defined in NPT, using the AEM and the BNPM. The mathematical models for relating the source of emissions to pollutant concentrations can be formulated using the influences of NPT. Each type of pollutant contributes the pollutant concentrations to a certain degree. The emission of pollutants can then be accumulated through the NPT cumulative influence. A model for spatial distribution of pollutants would also be required.

6.5.2 Chemical processes

NPT can also be extended for simulating chemical processes. Chemical processes are slightly different than nuclear physics processes as chemical processes allow reactions between two types of molecules, while a nuclear physics process, such as fission, involves a reaction between two types of nuclides. To represent chemical processes in NPT requires an additional framework for representing interactions between molecules performing chemical reactions. The framework must be capable of handling interactions between two or more atoms forming a molecule. It must also be capable of handling interactions between two or more molecules performing chemical reactions,

as well as representing the basic chemical reaction principles that follow the chemical reactions. As an example of a chemical reaction, two molecules of hydrogen react with two molecules of nitrogen oxide to form water and nitrogen is as follows:

$$2H_2 + 2NO \rightarrow 2H_2O + N_2$$

For chemical reactions, NPT requires the representation of the periodic table of the elements in a similar manner as for the representation of the nuclide data. All NPT influences, except for the decay influence, can still be used here.

6.5.3 Fusion

Fusion is the process of combining two light nuclear particles. The combined masses of fusion products are less than the mass of the original particles. The conversion of this tiny mass results in a huge torrent of energy. A kinetic energy equal to or greater than a potential energy of 0.29 MeV is required to enable nuclei to undergo fusion reactions against the Coulomb force [29]. A number of fusion reactions involving deuterium (^2_1H) are given below:

$$^2_1D + ^2_1D \rightarrow \begin{cases} ^3_2He + ^1_0n + 3.37 \text{ MeV}, \\ ^3_1T + ^1_1H + 4.03 \text{ MeV}, \end{cases} \tag{6.15}$$

$$^2_1D + ^3_1T \rightarrow ^4_2He + ^1_0n + 17.6MeV \tag{6.16}$$

Equations (6.15) and (6.16) represent D-D and D-T reactions respectively. Since there is no natural tritium ^3_1T [29], the ^3_1T can be produced via the reactions

$$^6_3Li + ^1_0n \rightarrow ^4_2He + ^3_1T + 4.8MeV \tag{6.17}$$

$$^7_3Li + ^1_0n \rightarrow ^4_2He + ^3_1T - 2.7MeV \tag{6.18}$$

NPT can be extended to represent fusion reactions, both for the D-D reaction and for the D-T reaction. There are two types of D-D reactions,

$$^2_1D + ^2_1D \rightarrow ^3_2He + ^1_0n + 3.27MeV \tag{6.19}$$

and

$$ {}^{2}_{1}D + {}^{2}_{1}D \rightarrow {}^{3}_{1}T + {}^{1}_{1}H + 4.03MeV, \tag{6.20} $$

respectively. Both ${}^{2}_{1}D$ and ${}^{3}_{2}He$ have been represented in the representation of the nuclide data, hence the D-D reaction can be represented in the NPT language in a manner similar to that of representing fission. The unstable nuclide ${}^{3}_{1}T$ must be added to the chart of nuclide data, and the representation of (6.15) via some distribution is also required. To enable nuclei to undergo a fusion reaction requires a kinetic energy equal or greater than 0.29 MeV, and therefore another field or influence must be added into the NPT framework to indicate there is a triggering force for fusion to occur.

Since tritium ${}^{3}_{1}T$ [29] is produced from either reaction (6.17) or (6.18), then to represent a D-T reaction , the reactions (6.17) and (6.18) must be represented in NPT accompanying the reaction written in equation (6.20).

6.5.4 Nuclear Medicine

Radiation in the form of gamma rays, beta particles, and neutrons in science and industry is used to achieve desirable changes. The use of radiation for medical therapy has greatly increased in recent years. The radiation comes from teletherapy units in which the source is at some distance from the target, or from isotopes in sealed containers implanted in the body or ingested in solutions of radionuclides. The main advantages of using radioisotopes are ease of detection of their presence through the emanations, and the uniqueness of the identifying halflives and radiation properties.

Different types of radionuclides used in medical diagnosis are reported in [1]. They include the gamma-ray emitter, e.g. Iodine-123 (${}^{123}I$) for thyroid therapy, Gallium-67 (${}^{67}Ga$) for tumor and abscess imaging, and Strontium-85 for bone scanning.

If a radioactive substance enters the body, radiation exposure to organs and tissues will occur. However, the foreign substance will not deliver all of its energy to the body

because of partial elimination. The effective time of irradiation is given by [28]

$$\frac{1}{T_{1/2\,EFFECTIVE}} = \frac{1}{T_{1/2\,PHYSICAL}} + \frac{1}{T_{1/2\,BIOLOGIC}}$$

where $\frac{1}{T_{1/2\,PHYSICAL}}$ and $\frac{1}{T_{1/2\,BIOLOGIC}}$ are the halflife of the radionuclide and the clearance half-time of residence in the organ, respectively.

For example, the radioactive iodine uptake study is performed by oral administration of a radioisotope of iodine ^{131}I to the fasting patient, followed by measurement of the uptake in the neck at an early interval varying from 2-6 hours and again at 24 hours. The emitted β-radiation is effective in irradiating the immediate local region of tissue in which ^{131}I is concentrated, since β-radiation is able to travel only millimeters in tissue [1].

To simulate the process of radiation absorption in tissue, one must know the amount of activity within the tissue, the total time of irradiation, and the number and abundance of emissions. One must know the geometry of the organ and the likelihood that radiation arising from decay in adjacent organs will reach the tissue of interest. The organ geometry is also important because it determines the volume in which the total amount of energy is deposited. The time-dose integral defines the amount of activity in the tissue, where the specific geometry determines the likelihood of a particle depositing all or part of its energy in the organ.

Since the greatest use of radionuclides for medical therapy is in the form of gamma and beta emitters, NPT can play a role in the simulation of the process of radiation in tissue in terms of representing the decay processes (the process of emitting gamma or beta particles) in the tissue. NPT cannot represent the whole activity within tissue, since it requires modeling of the organ geometry, modeling the molecules of the organ tissue, and modeling biological effects on the tissue.

6.5.5 Communication Networks

The distribution influences of NPT could also play a role in the simulation of computer communication networks. Different statistical distributions of traffic and noise could be modelled on network connections. The structure used for the chart of nuclides data could be used to represent network edges. A simulation of network traffic could then measure cumulative throughput on various edges and nodes in the network.

6.5.6 Software Validation

One of the important phases in software validation is module testing. The term module testing is used to encompass all procedures involving the testing of parts of a program in isolation. Module testing involves the testing of new untried code which can be presumed to include some faults and errors. It must seek to bring these faults to light so that they can be removed. The problem of ensuring that modules interface correctly can be broken into two components: physical and logical interfacing. At the physical level, a module may invoke another module and pass across a number of variables. It must be verified in some way that both modules assume that the same variables are passed across, that where necessary these variables are presented in the same order, that they include expected elements of data and that the elements of data are in the same expected order. At the logical level, it must be proven that if the called module assumes that a particular data item must be within a specific range, then the calling module provides a value in the expected range.

NPT can be adapted for representation of the process of statistical testing of modules; cumulative influences could be used for accumulating errors. A two-dimensional sparse array with rows and columns indicated calling and called modules, respectively, can be built using the chart of nuclide data structure. Each element (i, j) would contain the variables passed from module i to module j.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The work of this thesis gives a formal representation system for nuclear physics processes. As a result, Nuclear Physics Theory, NPT was established and was implemented. The results of the research have been met satisfactorily and the objectives outlined in section 1.2 have been met. In general the accomplishments can be stated as follows:

1. A formal representation of nuclear physics processes, NPT, was established which takes into account an accurate representation of nuclide data. It provides a proper way of expressing knowledge of nuclear physics processes.

2. A new formalism for representing non-deterministic influences (i.e. distribution influences) that affect physical processes has been devised, that can be applied to a wide variety of process modeling tasks.

3. NPT is a flexible framework which allows a user to express different types of nuclear physics processes.

4. NPT can describe interactions between highly dynamic processes in terms of influence variables in a process, or influence between variables in processes being modeled.

5. NPT can be used for reasoning about nuclear physics processes. The state of a process being modeled is described by a set of state space equations and a set of probabilistic space equations, which, when evaluated, provides the quantitative description of the processes in the current operating regime.

6. NPT can predict a behaviour of nuclear physics process quantitatively, not only by generating parameters that have quantities, but also by generating any type of both product nuclides and product particles which accompany almost any nuclear physics process.

7. NPT defines nuclear physics processes in terms of an easy-to-understand language that can express a wide variety of nuclear physics processes.

8. The methodology is easily extended to other nuclear physics processes.

7.2 Suggestions for future work

Possible extensions of this work include:

1. Develop a symbolic tool for nuclear physicists to allow them to write NPT expressions in an easy-to-use fashion.

2. Add more error checking in the syntax analyzer.

3. Add a display capability to the simulator that has been built so far so that the result of simulation can be graphically displayed for interpretation.

4. Add more realistic "machinery" to the neutron cycle. For example more detailed models of the moderator, cladding, control rods and the physical shape of the fuel would allow a more realistic simulation of an actual reactor.

5. Include a geometric model of reactor, such that it gives an accurate value of any reactor parameters in the model.

6. Extend the basic nuclear physics model and the aggregate effects model to allow them to better focus on those isotopes used in medical domain.

7. Explore the application of NPT in the medical domain.

8. Explore other nuclear physics simulations using NPT; e.g. delayed neutron fission, gamma decay, fusion. This will also require extending the current basic nuclear physics and aggregate effects model.

9. Investigate what would be required to speed up the simulation tool. Currently, a complete 20 second simulation requires about 4 hours to run on a SparcStation IPX.

10. Integrate a self-explanatory facility with NPT, such that it can provide causal accounts and characterize possible behaviour. This could become a core component in computer-based tutors.

11. Integrate NPT with work in virtual reality, such that visualization of any process interactions occur in real time. This could provide better explanations, and a better understanding of nuclear process interactions.

12. Implement NPTSim using parallel processing. Each processor would compute data for one nuclide (if there are at least 3000 processors) or a set of nuclides apportioned such that each processor handles an equal number of nuclides. This would speed up the simulation.

Bibliography

[1] Alazraki, N.P. and Mishkin, F.S. *Fundamental of Nuclear Medicine*, The Society of Nuclear Medicine, Inc, New York, NY, 1984.

[2] Bhaskar, R. and Nigam, A. "Qualitative Physics Using Dimensional Analysis", *Artificial Intelligence*, vol.45, 1990, pp.73-111.

[3] Bobrow, D. G. *Qualitative Reasoning about Physical Systems*, MIT Press, Cambridge, MA, 1985.

[4] Collins, J. W and Forbus, K.D. "Reasoning About Fluids Via Molecular Collections", *Proceedings of AAAI-1987*, pp.590-595.

[5] deKleer, J.D. "Multiple representations of knowledge in mechanics problem-solver", *Proceedings of IJCAI-77.*

[6] deKleer, J.D and Brown, J.S. "A qualitative physics based on confluences", *Artificial Intelligence*, vol.24, 1984, pp.7-83.

[7] deKleer, J.D and Bobrow, D.G. "A qualitative reasoning with high-order derivatives", *Proceedings of the National Conference on A.I.*, Austin, TX, Aug. 6-10, 1984, pp.86-91.

[8] Bobrow, D.G. "Qualitative Reasoning about Physical Systems: An Introduction", *Artificial Intelligence*, vol 24, 1984, pp.1-5.

[9] Dormoy, J. "Assembling a Device", *Proceedings AAAI-88 Seventh National Conference on Artificial Intelligence*, St Paul, Minnesota, USA, August 21-26, 1988, pp.330-335.

[10] Falting, B., and Struss, P. *Recent Advances in Qualitative Physics*, The MIT Press, Cambridge, Massachusetts, London, England, 1992.

[11] Forbus, K. D. *Qualitative Process Theory*, Ph.D. Thesis, Department of Computer Science, Massachusetts Institute of Technology, 1984.

[12] Forbus, K.D. "Introducing Actions into Qualitative Simulation", *Proceedings of the tenth International Joint Conferences on Artificial Intelligence*, 1989, pp.1273-1278.

[13] Foster, A. R. and Wright, R. L. "*Basic Nuclear Engineering* , Allyn and Bacon, Inc., Boston, MA, 1977.

[14] Hartati, S. "Reasoning about Physical Systems in Artificial Intelligence", *Technical Report TR93-080*, Faculty of Computer Science, University of New Brunswick, 1993.

[15] Hartati, S and Nickerson, B. G. "A Symbolic Model for Representing Nuclear Physics Processes", *APICS Annual Computer Science Conference Proceedings*, Wolfville, Nova Scotia, October 29, 1994, pp.99-108.

[16] Hartati, S and Nickerson, B. G. "Nuclear Process Theory", *Proceedings of the 11th IEEE Conference on AI Applications*, Los Angeles, CA, February 19-22, 1995, p340-346.

[17] Hartati, S. "Nuclear Process Theory: A Symbolic Model for Representing Nuclear Physics Processes", Technical Report TR94-087, Revision 2, Faculty of Computer Science, University of New Brunswick, May 4, 1995.

[18] Hartati, S and Nickerson, B.G. "An Efficient Computer Representation of Nuclide Data, *Computational Materials Science*, submitted June, 1995, 18 pages.

[19] Hartati, S., Nickerson, B.G. and DeMille, G.R. "Reasoning About Nuclear Physics Processes" *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence*, Washington, DC, November 5-8, 1995, pp.228-235.

[20] Hartati, S., Nickerson, B.G. and DeMille, G.R. "A model for simulation of nuclear physics processes", *International Journal of Modelling and Simulation*, accepted September, 1995, 24 pages (in press).

[21] Knief, R.A. "*Nuclear Energy Technology*", McGrawHill, NY, 1981.

[22] Kuipers, B. "Commonsense reasoning about causality: deriving behavior from structure", *Artificial Intelligence*, vol.24, 1984, pp.169-203.

[23] Kuipers, B. "Qualitative Simulation", *Artificial Intelligence*, vol.29, pp.289-388, 1986.

[24] Kuipers, B. and Berleant, D. "Using Incomplete Quantitative Knowledge in Qualitative Reasoning", in *Readings in Qualitative Reasoning about Physical Systems* edited by Weld, D and de Kleer, J, Morgan Kaufmann Publishers, Inc, San Mateo, California, 1990, pp.324-329.

[25] Lamarsh, John R. *Introduction to nuclear engineering*, Addison-Wesley, MA,1983.

[26] Lamarsh, John R. *Nuclear Reactor Theory*, Addison-Wesley, MA, 1966.

[27] Langhaar, H.L. *Dimensional Analysis and Theory of Models*, John Wiley and Sons Inc., NY, 1958.

[28] Palmer, E.L., Scott, J.A. and Struss, H.W. *Practical Nuclear Medicine*, W.B. Saunders Company, Philadelpia, Pennsylvania, 1992.

[29] Niu, K. "*Nuclear Fusion*, Cambridge University Press, NY, 1989.

[30] Jain, R. "*The art of computer systems performance Analysis Techniques for Experimental Design, Measurement, Simulation, and Modeling*", John Wiley and Sons, NY, 1991.

[31] Walker, F. W and Parrington J. R. "*Nuclides and Isotopes*", General Electric Company, San Jose, CA, 1989.

[32] Weiss, M. A. *Data Structures and Algorithm Analysis*, The Benjamin Cummings Publishing Company, Inc., Redwood City, CA, 1995.

[33] Weld, D and de Kleer, J. "*Readings in Qualitative Reasoning about Physical Systems*", Morgan Kaufmann Publishers, Inc, 1990.

[34] Weld, D. "*Theories of Comparative Analysis*", The MIT Press, Cambridge, Massachusetts, London, England, 1990.

[35] William, B. "Qualitative Analysis of MOS circuits", *Artificial Intelligence*, vol. 24, 1984, pp.281-346.

[36] Woods, E. "The Hybrid Phenomena Theory", *Proceedings of the 12th Int. Conference on A.I.*, Sydney, Australia, Aug. 24-30, 1991, pp.1138-1142.

[37] Woods, E. and Balchen, J.G. "Structural Estimation with the hybrid phenomena theory", *Proceedings of the 3rd IFAC Workshop*, California, Sep. 23-25, 1991, pp.127-132.

[38] Woods, E. *The Hybrid Phenomena Theory A Framework Integrating Structural Descriptions with State Space Modeling and Simulation*, Ph.D. Thesis, Division of Engineering Cybernetics, Norwegian Institute of Technology, Trondheim, Norway, 1993.

[39] "The Common Object Request Broker: Architecture and Specification", Sun Microsystems, Inc, 1992.

[40] Zannetti, P. *Environmental Modeling - Vol.I Computer Methods and Software for Simulating Environmental Pollution and its Adverse Effects*, Computational Mechanics Publications, Ashurts, Southhampton, UK, 1993.

Appendix I

Equations Used in NPT

This section gives an example of how to derive formulae for use in NPT for several types of nuclear processes. Since this is an example, this does not mean that equations obtained in this chapter are the only valid expressions for simulating nuclear processes. One could certainly choose a different set of equations to satisfy different needs. Since NPT evaluates state space equations every cycle, the equations to be derived have to be written based on this observation. It is not difficult to understand that selecting a reasonable time cycle is crucial in NPT. The equations used for this particular example are based on the neutron cycle of a reactor. To be more specific, they are based on the four factor equation [25]. For this reason, the average neutron life time in the reactor is selected as the NPT's cycle time period.

I.1 Fission processes in NPT

This section describes how to derive the four-factor formula [25] for use in NPT, which can describe the behavior of a thermal reactor system. It also describes how to generate state space models of neutron distribution, energy distribution of both gamma and neutron products, and the distribution of fission products. The four-factor formula components are :

1. fast fission factor, ϵ

2. resonance escape probability, p

3. thermal utilization factor, f

4. fission factor, η

Each of these has been described in section (3.2). These terms will be discussed in more detail in the next few sections.

I.1.1 Macroscopic cross section

The macroscopic cross section for a particular interaction of a mixture of substances is defined as the sum of the cross section for the particular interaction over all substances per unit volume of material. Thus, the macroscopic fission cross section of a mixture of n substances Σ_f^{fuel} is

$$\Sigma_f^{fuel} = N_1\,\sigma_f^1 + N_2\,\sigma_f^2 + \cdots + N_n\,\sigma_f^n. \tag{I.1}$$

where N_i, $\sigma_f^i (i = 1 \cdots n)$ represent the density of nuclides of type i and the fission cross section of nuclides of type i, respectively.

Similarly, the macroscopic absorption cross sections of a mixture of n substances Σ_a^{fuel} is

$$\Sigma_a^{fuel} = N_1\,\sigma_f^1 + N_2\,\sigma_f^2 + \cdots + N_n\,\sigma_f^n + N_1\,\sigma_g^1 + N_2\,\sigma_g^2 + \cdots + N_n\,\sigma_g^n. \tag{I.2}$$

where σ_g^i represents the radioactive capture cross section of nuclides of type i.

The above equations consist of two parts, which are notated by SP_1 and SP_2 in Fig. 5.2. The first n terms come from fission cross section of the n substances, while the second n terms come from the radioactive capture cross section of the n substances.

Each basic nuclear physics and aggregate effects model of a process is designed to represent interactions between neutrons and nuclides as a process (e.g. fission, radioactive capture). As a consequence, if more than one type of nuclide can instantiate a particular process, each type of nuclide will activate the basic nuclear physics and aggregate effects models individually. This gives NPT an ability to show the interaction between nuclides and neutrons individually. Once all types of nuclides have activated basic nuclear physics and aggregate effects models, the generated equations will show all interactions between neutrons and all types of nuclides and other processes.

To illustrate how to write the expression for macroscopic absorption cross section, consider the following example. Suppose we have two types of nuclides, type 1 and 2, as fuel in a reactor core. The nuclide of type 1 will activate basic nuclear physics and aggregate effects models of fission and produce an equation for macroscopic absorption cross section Σ_a^{fuel} as

$$\Sigma_a^{fuel} = N_1 \, \sigma_f^1.$$

The nuclides of type 2 also activate basic nuclear physics and aggregate effects models of fission. The corresponding term contributed by nuclides of type 2 is $N_2 \, \sigma_f^2$. These two terms are added by NPT to produce

$$\Sigma_a^{fuel} = N_1 \, \sigma_f^1 + N_2 \, \sigma_f^2. \tag{I.3}$$

The two types of nuclides, while activating basic nuclear physics and aggregate effects models of fission, also activate basic nuclear physics and aggregate effects models of radioactive capture and produce terms for macroscopic absorption cross sections as $N_1 \, \sigma_g^1$ and $N_2 \, \sigma_g^2$. These two terms are added by NPT to equation (I.3), which will result in an equation for macroscopic absorption cross section as

$$\Sigma_a^{fuel} = N_1 \, \sigma_f^1 + N_2 \, \sigma_f^2 + N_1 \, \sigma_g^1 + N_2 \, \sigma_g^2. \tag{I.4}$$

The more substances there are in the mixture, the more terms will appear in equation

(I.4). The generated equations are evaluated by the NPT evaluator to obtain the value of the LHS expressions, which in turn is used by the NPT simulator.

By looking at a particular equation in the state space model, one can tell what processes are active and what nuclides are involved in the processes. For example, by looking at equation (I.4) one can say that there are four types of interactions between neutrons and nuclides, i.e., interaction between neutrons and nuclides of type 1 through the fission process; between neutrons and nuclides of type 2 through the fission process; between neutrons and nuclides of type 1 through radioactive capture, and between neutrons and nuclides of type 2 through radioactive capture.

The macroscopic absorption cross section, Σ_a^{fuel}, is influenced by the fission process activated by nuclides of type j through $N^j\ \sigma_f^j$, and by the radioactive process through $N^j\ \sigma_g^j$. These influences are formalized in NPT using algebraic influences, i.e.

alg_infl { (Sigma_a_fuel)(obj1.amount,obj1.sigma_f)(obj1.amount*obj1.sigma_f) }
alg_infl { (Sigma_a_fuel)(obj1.amount,obj1.sigma_g)(obj1.amount*obj1.sigma_g) }

such as written in Figures 4.6 and 4.8 respectively.

I.1.2 Fission fraction

Initially, a certain number of neutrons along with one or more types of nuclides are given as an input; see e.g. the input description in Appendix II. These neutrons may interact with the nuclides in several possible ways; e.g. fission, radioactive capture. Assuming that we have four types of BNPM and AEM models in NPT, (i.e. fission, radioactive capture, beta positive and beta negative decays), neutrons can interact with nuclides through fission and radioactive capture only, since beta decay does not require neutrons. Assuming that n types of nuclides are available in the reactor core, and that there are X neutrons, the fraction of absorption of neutrons for each type of nuclide F_a^j will be

$$F_a^j = \frac{N_t\ \sigma_a^j}{N_1\ \sigma_a^1 + N_2\ \sigma_a^2 + \cdots + N_n\ \sigma_a^n}, \text{ for } j = 1 \cdots n,$$

where σ_a is the sum of the neutron absorption cross sections,

$$\sigma_a = \sigma_f + \sigma_g$$

and N_j is the density of nuclides of type j in the system. This equation shows that part of the neutrons, $F_a^j\ X$ interact with nuclides of type j through neutron absorption, which leads to both fissions and radioactive capture. Therefore the fraction of neutrons which interact with nuclides of type j through fission only , F_f^j, will be

$$F_f^j = \frac{N_j\ \sigma_f^j}{N_j\ \sigma_a^j} F_a^j,$$

which leads to

$$F_f^j = \frac{N_j\ \sigma_f^j}{N_1\ \sigma_a^1 + N_2\ \sigma_a^2 + \cdots + N_n\ \sigma_a^n}.$$

The fraction of neutrons which interact with nuclides of type j through radioactive capture, F_g^j, will be

$$F_g^j = \frac{N_j\ \sigma_g^j}{N_1\ \sigma_a^1 + N_2\ \sigma_a^2 + \cdots + N_n\ \sigma_a^n}.$$

The denominator of the last two equations is the macroscopic absorption cross section, which is denoted as Σ_a^{fuel}. Using this notation, the fraction of neutrons for fission and the fraction of neutrons for radioactive capture can be rewritten as

$$F_f^j = \frac{N_j\ \sigma_f^j}{\Sigma_a^{fuel}} \tag{I.5}$$

and

$$F_g^j = \frac{N_j\ \sigma_g^j}{\Sigma_a^{fuel}} \tag{I.6}$$

respectively. These equations show that only F_f^j of n_{th}^{fuel} neutrons will interact with nuclides of type j through fission, and F_g^j of n_{th}^{fuel} neutrons interact with the same type of nuclides through radioactive capture. The fraction of thermal neutrons absorbed in fuel, n_{th}^{fuel}, was discussed in Section 3.2.

When nuclides of type j having density N_j and fission cross section σ_f^j activates the fission process, it causes a fraction of neutrons, F_f^j, to interact with the nuclides. This influence is formalized by the NPT language through the algebraic influence

alg_infl { (Ff(obj1))(obj1.amount,obj1.sigma_f,Sigma_a_fuel)}

Similarly, when the nuclides of type j having capture cross section σ_j^g activate radioactive capture, it causes another fraction of neutrons, F_g^f, to interact with the nuclides. This influence is written in Fig. 4.8.

I.1.3 Thermal utilization factor

The thermal utilization factor, f, is defined as the ratio of the number of thermal neutrons absorbed in the fuel to the total number of absorptions in the fuel, cladding, moderator and other materials [21], i.e.,

$$f = \frac{\Sigma_a^{fuel}}{\Sigma_a^{fuel} + \Sigma_a^{non-fuel}}. \tag{I.7}$$

If the fuel is a mixture of n substances, Σ_a^{fuel} is given by equation (I.2). Substituting equation (I.2) into the denominator of equation (I.7) results in

$$f = \frac{N_1\,\sigma_f^1 + N_2\,\sigma_f^2 + \cdots + N_n\,\sigma_f^n + N_1\,\sigma_g^1 + N_2\,\sigma_g^2 + \cdots + N_n\,\sigma_g^n}{\Sigma_a^{fuel} + \Sigma_a^{non-fuel}} \tag{I.8}$$

which can be written as

$$f = \frac{N_1\,\sigma_f^1 + N_1\,\sigma_g^1}{\Sigma_a^{fuel} + \Sigma_a^{non-fuel}} + \frac{N_2\,\sigma_f^2 + N_2\,\sigma_g^2}{\Sigma_a^{fuel} + \Sigma_a^{non-fuel}} + \cdots + \frac{N_n\,\sigma_f^n + N_n\,\sigma_g^n}{\Sigma_a^{fuel} + \Sigma_a^{non-fuel}} \tag{I.9}$$

or more compactly as

$$f = f_1 + f_2 + \cdots + f_n.$$

Since there are n substances involved in both fission and radioactive capture, each substance contributes

$$f_j = \frac{N_j\,\sigma_f^j + N_j\,\sigma_g^j}{\Sigma_a^{fuel} + \Sigma_a^{non-fuel}}, \text{ for } j = 1, 2, \cdots, n \tag{I.10}$$

to f. If the non-fuel contributors are only cladding and moderator, then $\Sigma_a^{non-fuel}$ is given by

$$\Sigma_a^{non-fuel} = \Sigma_a^{cl} + \Sigma_a^{md}$$

where Σ_a^{cl} and Σ_a^{md} are the macroscopic absorption cross section for cladding and moderator, respectively.

If only one type of nuclide activates both basic nuclear physics and aggregate effects models of fission and radioactive capture, Σ_a^{fuel} reduces to

$$\Sigma_a^{fuel} = N\,\sigma_f + N\,\sigma_g.$$

Since only one type of nuclide actives both basic and aggregate effects models of fission at a time, the thermal utilization factor is given by

$$f = \frac{N\,\sigma_f + N\,\sigma_g}{\Sigma_a^{fuel} + \Sigma_a^{cl} + \Sigma_a^{md}}. \tag{I.11}$$

The thermal utilization factor f is formalized by the NPT language through an algebraic influence such as

alg_infl { (f(obj1))

(obj1.amount,obj1.sigma_f,obj1.sigma_g,Sigma_a_fuel,Zra.sigma_a,HW.sigma_a)}

I.1.4 Thermal fission factor

The thermal fission factor, η, is defined as the ratio of the number of fast neutrons produced per thermal neutron absorbed in fuel [13], i.e.

$$\eta = \frac{\nu\,\Sigma_f^{fuel}}{\Sigma_a^{fuel}}.$$

The general expression for the thermal fission factor of a mixture of n substances is

$$\eta = \frac{\nu_1\,N_1\,\sigma_f^1 + \nu_2\,N_2\,\sigma_f^2 + \cdots + \nu_n\,N_n\,\sigma_f^n}{\Sigma_a^{fuel}} \tag{I.12}$$

which can be written as

$$\eta = \frac{\nu_1\,N_1\,\sigma_f^1}{\Sigma_a^{fuel}} + \frac{\nu_2\,N_2\,\sigma_f^2}{\Sigma_a^{fuel}} + \cdots + \frac{\nu_n\,N_n\,\sigma_f^n}{\Sigma_a^{fuel}} \tag{I.13}$$

where ν_j is the average fission neutrons produced from fissioning nuclide of type j. The values of ν_j of all different types of nuclides are not completely known, and only

a few of nuclides have a known value of ν, (it is approximately 2.43). Assuming that the values of ν for different types of nuclides are the same (e.g. $\nu = 2.43$), equation (I.13) becomes

$$\eta = \nu \left(\frac{N_1 \sigma_f^1}{\Sigma_a^{fuel}} + \frac{N_2 \sigma_f^2}{\Sigma_a^{fuel}} + \cdots + \frac{N_n \sigma_f^n}{\Sigma_a^{fuel}} \right) \tag{I.14}$$

or

$$\eta = \eta_1 + \eta_2 + \cdots + \eta_n$$

Each term $\eta_j (j = 1 \cdots n)$ represents the thermal fission factor as a result of fissioning nuclides of type j. When the nuclides of type j activate the fission process, η is affected by the process through function

$$\frac{\nu \ N_j \ \sigma_f^j}{\Sigma_a^{fuel}} \tag{I.15}$$

This influence is formalized in NPT language through the algebraic influence

alg_infl { (eta)(nu,obj1.amount,obj1.sigma_f,Sigma_a_fuel)
(nu*obj1.amount*obj1.sigma_f/Sigma_a_fuel) }

I.1.5 Multiplication factor

The multiplication factor k is defined as the ratio of the number of neutrons in the current generation, n_{g+1}, to the number of neutrons in the previous generation, n_g; i.e.

$$k = \frac{n_g}{n_{g-1}}.$$

The multiplication factor can be expressed in terms of the four-factor formula [25] as

$$k = \eta \ \epsilon \ p \ f \ P_{NLf} P_{NLth}. \tag{I.16}$$

Since BNPM and AEM models of fission can be instantiated by one type of nuclide at a time, then if only one type of nuclide is available for the fission process, k is given by equation (I.16). If n types of nuclides activate both BNPM and AEM models of

fission and radioactive capture, there will be a state space equation for each f_j and η_j as discussed before (see equations (I.15) and (I.10)). Each type of nuclide contributes a term to k. The contribution is

$$\eta \in p\ f_j\ P_{NLf}\ P_{NLth}$$

The equation of k for more then one type of nuclide involved, is defined as

$$k = \eta \in p\ f_1\ P_{NLf}\ P_{NLth} + \eta \in p\ f_2\ P_{NLf}\ P_{NLth} + \cdots + \eta \in p\ f_n\ P_{NLf}\ P_{NLth}$$

or

$$k = k_1 + k_2 + \cdots + k_n$$

where $k_j (j = 1 \cdots n)$ represents the contribution of nuclide j to the multiplication factor.

The influence on the multiplication factor k is formalized in NPT as the algebraic influence

alg_infl {(k)(f(obj1),eta,
PHWNUR.epsilon,PHWNUR.p,PHWNUR.PNLf,PHWNUR.PNLth)
(f(obj1)*eta*PHWNUR.epsilon*PHWNUR.p*PHWNUR.PNLf*PHWNUR.PNLth) }

I.1.6 Fissions per cycle

As discussed in section (3.2), the NPT model evaluates the number of fissions every prompt neutron life time, l_p, which is called the cycle time, t_c. The number of fissions in the present cycle as a result of fissioning certain types of nuclides is given by equation (3.5), R_{c_g}, and can be rewritten as

$$R_{c_g} = \frac{k}{\eta}\ X\ F_f$$

where X is the total number of fission neutrons in the previous generation , and F_f is the fission fraction.

As explained in the previous sections, each type of fissioning nuclide gives rise to the corresponding equations for k, η and F_f^j. If the fuel is composed of n substances, and X neutrons are generated in the previous generation, the number of fissions at the present cycle caused by each substance is

$$R_{c_g}^j = \frac{k}{\eta} F^j X \tag{I.17}$$

For brevity, R_{c_g} will be referred to by R_c in the succeeding discussion. Equation (I.17) can be rewritten as

$$R_c^j = \frac{k}{\eta_j} F^j X. \tag{I.18}$$

The above equation represents the number of fissions at the present cycle involving nuclides of type j. When the nuclides of type j activate a fission process, it causes equations for k, η and F^j to exist, which in turn, give rise to equation I.18. This influence is expressed in the NPT language through algebraic influence such as

alg_infl {(Rc(obj1))(k,eta,obj2.amount,Ff(obj1))(k/eta*obj2.amount*Ff(obj1))}

where obj2.amount is X in equation (I.18), and obj1 is symbolic expression for the activating nuclides.

I.1.7 Fission products

There is a set of products associated with each fission process. The products of the fission process described in this section are fission neutrons, energy of fission neutrons, product nuclides, energy of product nuclides, and energy of gamma particles.

I.1.7.1 Fission neutrons

For each active process, NPT generates equations for the number of fissions per cycle for each type of nuclide, R_c^j. Fission neutrons are produced when neutrons interact with nuclides under fission processes. The number of fission neutrons is probabilistic. The probability of producing 0-8 neutrons is controlled by the Poisson distribution.

This distribution is represented by pdf_neutron. Once the equation of the number of fissions caused by each type of nuclide is generated, the equation for the distribution of fission neutrons for each type of nuclide, $distr\_neutron[i]^j$ (see Fig. 4.6), can be defined; i.e.

$$distr\_neutron[i]^j = R_c^j * pdf\_neutron[i],\ i = 0..8. \tag{I.19}$$

The superscript j represents the type of nuclide activating the process under consideration. The equation for the total number of fission neutrons generated by nuclides of type j pn^j is

$$pn^j\ = sum(i * distr\_neutron^j[i], i = 0..8). \tag{I.20}$$

When nuclides of type j activate the fission process, an equation for the number of fissions per cycle, R_c^j, is produced. This will influence the distribution of fission neutrons, and the influence is formalized by the NPT language using the aggregate influence. In the NPT language the above equation becomes

alg_infl { (pn(obj1))(distr_neutron(obj1)) (sum(i*distr_neutron(obj1)[i],i=0..8)) }

I.1.7.2 Product Nuclides

The mass numbers of product nuclides are probabilistic. For example, for ^{235}U the mass number ranges from 73 - 162. Fission always produces a pair of nuclides, F1 (light nuclide) and F2 (heavy nuclide), where the mass number of F1 typically ranges between 72 - 117, while F2 is from 118 to 162. The range of mass numbers of product nuclides varies from nuclide to nuclide.

The probability of having a certain mass number is typically expressed in terms of % mass yield distribution of the fissioning nuclides. Part of the mass yield distribution for ^{235}U is shown in Table I.1. In our example, mass yield probabilities for F1 and F2 are represented by pdf_A1 and pdf_A2, respectively. The distribution of the mass numbers of product nuclides are given by distr_A1 for F1 and distr_A2 for F2.

Table I.1: Probability of fission fragments having a certain mass number for activating nuclide U235.

A1	% yield	A1	% yield	A2	% yield	A2	% yield
	pdf_A1		pdf_A1		pdf_A2		pdf_A2
72	0.000026	95	6.50	118	0.013	142	6.21
73	0.0001	96	6.50	119	0.011	143	6.58
74	0.0003	97	6.3	120	0.013	144	6.584
75	0.001	98	5.98	121	0.013	145	6.595
76	0.003	99	5.78	123	0.013	146	5.50
77	0.008	100	6.1	124	0.016	147	3.93
78	0.021	101	6.28	125	0.0159	148	3.00
79	0.044	102	5.18	126	0.027	149	2.25
80	0.13	103	4.29	127	0.031	150	1.67
81	0.13	104	3.03	128	0.059	151	1.08
82	0.19	105	1.88	129	0.126	152	0.653
83	0.32	106	0.96	130	0.35	153	0.417
84	0.535	107	0.41	131	0.75	154	0.268
85	1.00	108	0.145	132	1.81	155	0.158
86	1.317	109	0.054	133	2.89	156	0.074
87	1.96	110	0.031	134	4.31	157	0.032
88	2.55	111	0.025	135	6.69	158	0.0149
89	3.57	112	0.019	136	7.87	159	0.0062
90	4.76	113	0.013	137	6.54	160	0.0033
91	5.8	114	0.015	138	6.32	161	0.0010
92	5.84	115	0.013	139	6.19	162	0.0003
93	6.03	119	0.0096	140	6.71	163	0.0085
94	6.37	117	0.00106	141	6.4		

As discussed before, an active fission process produces equations for the number of fissions per cycle for each type of nuclide, R_c^j. Once this state space equation is created, the state space equation for distribution of mass numbers of fission products of F1 and F2 will be

$$distr\_A1[i]^j = R_c^j * pdf\_A1^j[i],\ i = lb_j..\frac{ub_j}{2} \tag{I.21}$$

$$distr\_A2[i]^j = R_c^j * pdf\_A1^j[i],\ i = \frac{ub_j}{2}..ub_j \tag{I.22}$$

respectively, where lb and ub are the lower and upper bound range of mass numbers of fission products. lb and ub vary from nuclide to nuclide.

Once the nuclides of type j activate a fission process, the equation for R_c^j is generated, which influences the distribution of mass numbers of fission products of F1 and F2 such as written in equations (I.21) and (I.22). These influences are expressed in the NPT language through the aggregate influence, one of them is shown here such as

aggr_infl { (distr_A1(obj1)[i])(Rc(obj1),pdf_A1) (Rc(obj1)*pdf_A1[i],i=73..117) }

In order to know what the fission products are, NPT has to provide a way to generate atomic numbers from the known mass number. Consider a nuclide with mass number A. The associated Z can be approximated by *proton − neutron ratio* [25] with uncertainty δ_Z, as

$$Z = \frac{Z_c}{A_c - \nu} A + \delta_Z \tag{I.23}$$

where ν is the average product neutrons per fission, A_c and Z_c represent the mass number and the atomic number of the compound nucleus, respectively. The term δ_Z represents the deviation of product nuclide (Z, A) from the stable zone of nuclides, and it has a non-deterministic value. The distribution of δ_Z can be approximated by the Gaussian distribution. Given a number of product nuclides with a certain mass number A, δ_Z distributes these nuclides to produce many nuclides with different atomic numbers (isobars).

Once the distribution of mass numbers, distr_A1 , is known, its element consists of the number of product nuclides having the same mass numbers given by the element's index (isobars). The NPT language provides a way for decomposing the isobars into different types of nuclides having different atomic numbers. Each type of nuclide has unique atomic numbers F1.Z and neutron numbers F1.N

Knowing distr_A1 and pdf_dZ, each element in distr_A1 represent a set of isobars of product nuclides F1 with mass numbers given by the associated index. To identify how many types of nuclides occur within each element of distr_A1 , several tasks are carried out

1. Determine a common isobars' atomic numbers using *proton neutron ratio* $Z = \frac{Z_c}{A_c - \nu} A$ (see equation (I.23)).

2. Determine the atomic number of each type of nuclide F1.Z, by giving a deviation δ_Z, represented by an index ofpdf_dZ, to the value of item 1.

3. Determine the neutron number of each type of nuclides F1.Z, by subtracting the known mass numbers with the value of item 2.

4. Since the number of isobars in each element of distr_A1 are distributed over pdf_dZ, the number of each type of nuclide in the isobars,F1.amount, is calculated by multiplying every element of distr_A1 with elements of pdf_dZ. If the result is converted into grams, it is multiplied by Avogadro's number, then divided by the associated mass number.

5. The above processes are repeated for all set of isobars represented in distr_A1 . Similarly, these process are carried out for all isobars represented in distr_A2 .

These above processes are formalized in the NPT language using the distribution influence for fission fragments F1 such as for fission fragments F1 such as

distr_infl { (F1)(distr_A1,pdf_dZ)

(F1.Z=c_n.Z/(c_n.A-nu)*A+dZ;

F1.N=A-(c_n.Z/(c_n.A-nu)*A+dZ);

F1.amount=distr_A1(obj1)[A]*pdf_dZ[dZ]/NA*A, dZ=-4..4,A=73..117) }

Therefore, when a type of nuclide activates a fission process, the process gives rise to distr_A1 and pdf_dZ, which in turn cause product nuclides F1 (having different type of nuclides) and F2 (having different types of nuclides) to be produced. Both c_n.A,c_n.Z and nu are the same as A_c and Z_c and ν in equation (I.23), respectively.

I.1.7.3 Energy of fission neutrons

Prompt neutrons are emitted with a continuous distribution of energies called the *prompt-neutron-spectrum* [26]. Equation (I.24) gives the distribution of this *prompt-neutron spectrum*

$$\chi(E) = 0.453e^{1.036E} sinh\sqrt{2.29E} \tag{I.24}$$

where $\chi(E)dE$ is the number of fission neutrons emitted with energy between E and $E + dE$.

The probability of a prompt neutron having an energy in a certain interval $[e_1, e_2]$ is obtained by calculating the area under the *prompt-neutron spectrum* which is bounded by e_1 and e_2. Examples of the probabilities of a neutron having an energy in a certain interval are tabularized in Table I.2. pdf_E_neutron in the probabilistic space equations contains probability values given by the prompt-neutron spectrum, part of which are shown in Table I.2.

The total energy of neutron products for a number of fissions is accumulated such as follows

$$\sum_{i=0}^{8}(i + ei/2) * pdf\_E\_neutron[i] \tag{I.25}$$

where ei is an energy interval in pdf_E_neutron and $i + 0.5$ is the average energy of the i^{th} energy interval of neutron products.

Table I.2: Probability per unit interval of energy of neutron products calculated from [13].

neutron energy En	*probability of producing En*, pdf_E_neutron[i]	i
[0, 1]	0.3101635382	0
[1, 2]	0.2965603757	1
[2, 3]	0.1858727779	2
[3, 4]	0.1033541111	3
[4, 5]	0.0538716708	4
[5, 6]	0.0269146713	5
[6, 7]	0.0130406600	6
[7, 8]	0.0061712412	7
[8, 9]	0.0028658063	8

Once the probabilistic space equation of the distribution of fission neutrons, and the state space equation of the number of fission neutrons are generated by NPT (such as given by equations (I.19) and (I.20)), NPT generates equations for energy distribution of fission neutrons such as

$$distr\_E\_neutron[i]^j = pn^j * pdf\_E\_neutron^j[i], \text{ for } i = 0..8$$

where the superscript j indicates the fissioning nuclides. The equation for the energy of fission neutrons as a result of interaction between neutrons and nuclides of type j is accumulated as

$$Neutron.energy \mathrel{+}= (i + ei/2) * distr\_E\_neutron^j[i], \text{ for } i = 0..8 \qquad \text{(I.26)}$$

The "+=" operator in equation (I.26) is used to denote an accumulation process. This equation shows that *Neutron.energy* is obtained by accumulating the RHS of equation (I.26) resulting from different types of nuclides.

The generation of the above equation is formalized by the NPT language through a cumulative influence, for example

cum_infl { (Neutron.energy)(distr_E_neutron(obj1))

(sum((i+ei/2)*distr_E_neutron(obj1)[i],i=0..8)) }

I.1.7.4 Energy of gamma particles

Gamma particles are emitted with a continuous distribution of energies which is called the *prompt-gamma-spectrum* [26]. Equation (I.27) gives the distribution of *prompt-gamma spectrum*

$$N(E) = e^{1.10E} \tag{I.27}$$

where $N(E)dE$ is the number of γ-rays emitted with gamma an energy between E and $E + dE$. .

The probability of a prompt gamma having an energy in a certain interval $dE = [e_1, e_2]$ is obtained by calculating the area under the *prompt-gamma spectrum* which is bounded by e_1 and e_2. Examples of the probabilities of a gamma having an energy in a certain interval are shown in Table I.3. pdf_E_gamma in the probabilistic space model contains probability values given by the prompt-gamma spectrum, part of which are shown in Table I.3.

Table I.3: Probability per unit interval of energy of gamma products.

gamma energy Eg	*probability of producing* Eg, pdf_E_gamma[i]	i
[0, 1]	0.666481	0
[1, 2]	0.221880	1
[2, 3]	0.077200	2
[3, 4]	0.022369	3
[4, 5]	0.007446	4
[5, 6]	0.002479	5
[6, 7]	0.000825	6
[7, 8]	0.000275	7
[8, 9]	0.000091	8

Having pdf_E_gamma defined, the equation for the gamma energy distribution

caused by interaction of neutrons and nuclides of type j is given by

$$distr\_E\_gamma[i]^j = R_c^j * pdf\_E\_gamma^j[i], \text{ for } i = 0..8.$$

The energy of gamma products, as a result of interactions between neutrons and nuclides of type j, is accumulated as

$$Gamma.energy + = (i + ei/2) * distr\_E\_gamma^j[i], \text{ for } i = 0..8.$$

The generation of the above equation is formalized by the NPT language through a cumulative influence, for example

cum_infl { (Gamma.energy)(distr_E_gamma(obj1))
(sum((i+ei/2)*distr_E_gamma(obj1)[i],i=0..8)) }

I.1.7.5 Energy of product nuclides

The major fraction of the energy released in a fission process appears as kinetic energy of the two fission fragments (F1 and F2). The distribution of mass numbers of fission fragments and the distribution of kinetic energy of the fission fragments are related by equations described below. For the two fragments (designated F1 and F2) in a binary fission, the momenta of F1 and F2 must be equal, i.e.

$$M_1 v_1 = M_2 v_2.$$

Since

$$E_1 = \frac{1}{2} M_1 v_1^2 \text{ and } E_2 = \frac{1}{2} M_2 v_2^2$$

therefore

$$\frac{E_1}{E_2} = \frac{M_2}{M_1}.$$

Since the mass distribution of fission products are known, the energy distribution of fission products can be obtained using the above relations. In our example, the energy distribution for the two product nuclides F1 and F2 are represented by pdf_E_F1

and pdf_E_F2, respectively (see Fig. 4.2). Having pdf_E_F1 defined, the equation for the energy distribution of fission products F1 caused by interaction of neutrons and nuclides of type j is given by

$$distr\_E\_F1[i]^j = R_c^j * pdf\_E\_F1^j[i], \text{ for } i = 0..8.$$

The energy of fission products F1, as a result of interactions between neutrons and nuclides of type j, is accumulated as

$$Energy\_F1.amount^j \mathrel{+}= (i + ei/2) * distr\_E\_F1^j[i], \text{ for } i = 0..8.$$

The generation of the above equation is formalized by the NPT language through another cumulative influence, i.e.

cum_infl { (Energy_F1.amount)(distr_E_F1(obj1))
(sum((i+ei/2)*distr_E_F1(obj1)[i],i=40..84)) }

I.1.8 Fuel Consumption

NPT evaluates changes in parameters every cycle time. Every product generated or object consumed in an active process is evaluated every cycle time. Every active process in NPT evaluates its fuel consumption directly. If there are m types of processes activated by one type of nuclide, then there will be m parts for fuel consumption, and each is consumed differently depending on the process.

For example, in the fission process NPT evaluates the number of fissions per cycle time, and the fuel consumption for fission per cycle for a certain type of nuclide, C_f^j

$$C_f^j = \frac{R_c}{NA} A$$

where NA is Avogadro's number, and A is the mass number of the activating nuclide. Note that C_f is in units of grams. Consumption of nuclides of type j for fission processes is expressed in the form of

$$j.amount \mathrel{+}= -C_f^j.$$

If this type of nuclide also activates the basic and aggregate effects models for radioactive capture, then the equations for fuel consumption also reflect this fact. Denoting C^j_{rc} as the amount of nuclides of type j consumed by radioactive processes (see section I.4), the equation for the consumption of this type of nuclide will be of the form

$$j.amount \mathrel{+}= -C^j_f - C^j_{rc}.$$

Similarly, if C^j_{bnd} represents the amount of nuclides of type j consumed for beta negative decay processes, the form of the equation for consumption of nuclide j for fission, radioactive capture and beta negative decay processes will be

$$j.amount \mathrel{+}= -C^j_f - C^j_{rc} - C^j_{bnd}.$$

The details of C^j_{rc} and C^j_{bnd} are given in the next few sections.

I.2 Beta negative decay in NPT

The beta negative decay process occurs for nuclei in which there is an excess of neutrons. In this process, a neutron (${}^1_0 n$) is considered to be converted into a proton (${}^1_1 p$) with emission of an electron (${}^0_{-1}e$) or β^-. The nuclear charge, Z, is consequently increased by one with the mass number being unchanged. This is indicated by

$${}^A_Z T \xrightarrow{\beta^-} {}^A_{Z+1} T. \tag{I.28}$$

The nuclide ${}^A_{Z+1}T$ is regarded as a new nuclide, and can be denoted as ${}^A_Z P$.

Equation (I.28) shows that a radioactive species T decays into another species P by emitting β^-. The equations for the rate of change of the number of atoms with time for each species are given by

$$\frac{dN_T}{dt} = -\lambda_T \, N_T, \tag{I.29}$$

$$\frac{dN_P}{dt} = \lambda_T \, N_T. \tag{I.30}$$

Equation (I.29) shows a consumption of the number of atoms of the decaying nuclides over time, while equation (I.30) shows the production of the resulting nuclide P over time. The change in the number of atoms of the decaying nuclides at a very small time period t_c, can be approximated by

$$dN_T = -\lambda_T \; N_T \; t_c. \tag{I.31}$$

The amount of production of the resulting nuclide for a certain time period is the same as the amount of consumption for that period, i.e

$$dN_R = \lambda_T N_T t_c,$$

where P is the resulting nuclide.

In our example, the equation for nuclide consumption due to the beta negative decay process, C^T_{bnd}, is written in the form

$$C^T_{bnd} = \lambda_T \; N_T \; t_c \tag{I.32}$$

while the resulting nuclide, P^P_{bnd}, is given by

$$P^P_{bnd} = \lambda_T \; N_T \; t_c. \tag{I.33}$$

Some equations that are also involved in the beta decay process and generated every cycle time are equations (I.32), (I.33) and

$$T(Z,A).amount - = \; C^T_{bnd}, \tag{I.34}$$

and

$$P(Z,A).amount + = \; P^P_{bnd}. \tag{I.35}$$

Equations (I.34) and (I.35) show the the number of nuclides T decreases by the value of C^T_{bnd}, and the number of nuclides P increases by the value of P^P_{bnd}.

When a beta negative decay occurs, it emits a beta particle with the corresponding energy release of 0.4 MeV [25]. To show this process, the equation for the number

of beta particles is generated as well as the amount of energy of beta particles. The number of beta particles released is equal to the number of product nuclides P. The total number of beta negative particles from decaying a particular type of nuclide, $Beta\_Neg.amount$, is given by

$$Beta\_Neg.amount \mathrel{+}= P_{bnd}^{P} * 1. \tag{I.36}$$

The corresponding equation for the amount of energy of beta particle, $Beta\_Neg.energy$, is given by

$$Beta\_Neg.energy \mathrel{+}= P_{bnd}^{P} * 0.4 \tag{I.37}$$

Each basic and aggregate effects model in NPT is activated by one type of nuclide at a time. Therefore, when the basic nuclear physics and aggregate effects model of beta negative decay is activated by a type of nuclide, the resulting equation for nuclide consumption is given by equation (I.32). The equation for nuclide production is given by equation (I.33).

If the activating nuclide has two beta decay modes, β^- and β^+, the resulting equation for the nuclide consumption will be from both β^- and β^+ processes. The one which comes from beta negative decay will be

$$C_{bnd}^{P} = 0.5\lambda_T \; N_T \; t_c \tag{I.38}$$

with the corresponding state space equation for P given by

$$P_{bnd}^{P} = 0.5\lambda_T \; N_T \; t_c. \tag{I.39}$$

In this case the amount of production and consumption associated with the beta positive decay process is generated by the aggregate effects model of beta positive decay.

The generation of equations for the product nuclides of beta negative decay, either equation (I.39) or (I.33), is formalized in the NPT language using decay influences as follows:

```
decay_infl { (obj1.dmode & BETA_POSITIVE)(Pbnd(obj1))
  (ld,obj1.halflife,obj1.amount,NA,obj1.A,cycle_time)
  (0.5*ld*obj1.halflife*obj1.amount*NA/obj1.A*cycle_time) }

decay_infl { (!(obj1.dmode & BETA_POSITIVE))(Pbnd(obj1))
  (ld,obj1.halflife,obj1.amount,NA,obj1.A,cycle_time)
  (ld*obj1.halflife*obj1.amount*NA/obj1.A*cycle_time) }
```

I.3 Beta positive decay in NPT

Beta positive decay occurs for a nuclide in which there is a deficiency of neutrons. In this process, a proton (${}^{0}_{+1}\ p$) is considered to be converted into a neutron with emission of a positron (${}^{0}_{+1}\ e$) or β^+. The resultant nucleus has a nuclear charge of the originating nuclear charge minus one, with the mass number being unchanged. This is denoted by

$$ {}^{A}_{Z}T \xrightarrow{\beta^+} {}^{A}_{Z-1}T. $$

The nuclide ${}^{A}_{Z-1}T$ can be denoted as a new nuclide ${}^{A}_{Z}P$.

Since the beta positive decay process is very similar to the beta negative decay process, the basic nuclear physics and aggregate effects models of the two processes are very similar. Consequently, when these models are activated by a nuclide the state space equations generated for the beta positive decay processes are very similar. The state space equations for a beta positive decay process are similar to equations (I.32), (I.33), (I.34), (I.35), (I.36) and (I.37), except that they have different LHS expressions. Similarly, the same influences as those used in the beta negative decay are employed in this process.

I.4 Radioactive capture in NPT

Radioactive capture processes occur when a neutron is captured by a nucleus and a γ particle is emitted. In general, the radioactive capture process can be expressed as

$$ {}^{A}_{Z}T + {}^{1}_{0}n \longrightarrow {}^{A+1}_{Z} T, $$

where the nuclide ${}^{A+1}_{Z}T$ can be denoted as a new nuclide ${}^{A}_{Z}P$.

As discussed before, there are $X\ \frac{k}{\eta}$ thermal neutrons available in the reactor core. There are also n types of nuclides involved in the reactor core. These thermal neutrons interact with the n types of nuclides through several different processes.

The fraction of neutrons interacting through radioactive capture for each type of nuclide (e.g. nuclide of type j) is given by equation (I.6). At the current cycle, the production of P as a result of interactions between neutrons and nuclides of type j which cause a radioactive capture process is given by R^j_{rc}, where

$$ R^j_{rc} = F^j_g\ X\ \frac{k}{\eta}. \tag{I.40} $$

The consumption of T (nuclides of type j) for radioactive capture is equal to equation (I.40), since one nuclide T is consumed for every nuclide P produced.

This gives C^j_{rc} as follows

$$ C^j_{rc} = F^j_g\ X\ \frac{k}{\eta}. $$

Consequently, C^j_{rc} decreases the amount of nuclides T, and R^j_{rc} increases the amount of nuclides P. The amount of T decreases by the amount of consumption of nuclides T during a cycle time, that is

$$ T(Z,A).amount - = C^j_{rc}, $$

where $T(Z,A)$ is the actual activating nuclide ID. This task is specified through the NPT language as a cumulative influence

cum_infl { (obj1.amount)(Crc(obj1))(-Crc(obj1)) }

where obj1 will be replaced with the actual decaying nuclide ID.

The equation for the number of resulting nuclides P, $P(Z,A).amount$, is given by

$$P(Z,A).amount \mathrel{+}= P^{j}_{rc}$$

where $P(Z,A)$ is the actual resulting nuclide ID.

Appendix II

NPTsim Input descriptions for Experiment 2

The experiment is made to model nuclear physics process in a Pressurized Heavy Water Natural Uranium reactor core (PHWNUR). In this model, the core was fueled by a Natural Uranium which consists of 0.711 % of ^{235}U and 99.289 % of ^{238}U. The moderator used in the Pressurized Heavy Water Natural Uranium is heavy water and zircaloy was used as cladding.

```
simulation_time 20.00 s
report_time 1.00 s
cycle_time 0.02 s

report Fissile_fissioned          report Fissile_consumed
report Energy_F1.amount           report Energy_F2.amount
report Neutron.amount             report Neutron.energy
report Gamma.energy               report Beta_Positive.amount
report Beta_Positive.energy       report Beta_Negative.amount
report Beta_Negative.energy       report U235.amount
report U238.amount


NUCLIDE U235 {
    Z 92
    N 143
    amount 0.7e+5 }
NUCLIDE U238 {
    Z 92
    N 146
    amount 99.3 e+5 }
PARTICLE neutron {
    id 32
    amount 1e+20 }
REACTOR PHWNUR {
    power ON
    epsilon 1.03
    p 0.667
    PNLth 0.975
    PNLf 0.985 }
MODERATOR HW {
    sigma_a 0.001
    density 3.006e+22 }
CLADDING Zra {
    sigma_a 0.185
    density 2.013e+23 }
```

Figure II.1: NPTsim input descriptions of experiment no.2.

Appendix III

Formal Grammar of the NPT language

This appendix defines the grammar for the input files of the current NPT language and its implementation. Some of the notation used is adapted from [39]. The symbols used are defined below.

Notation	
::=	Is defined to be
|	Alternatively
<text>	Non-terminal
"text"	Literal is always between double quotes
*	The preceding syntactic unit can be repeated zero or more times
**	The preceding syntactic unit can be repeated zero or one time
+	The preceding syntactic unit can be repeated one or more times
{}	The enclosed syntactic unit are grouped as a single syntatic unit
/♯...♯/	Encloses a comment in the NPT grammar definition

```
<number>            ::= <real> | <integer>
<real>              ::= <dreal> | <ereal>
<integer>           ::= <0-9>+  | <0-9>+<eE><0-9>+
<dreal>             ::= <0-9>*"."<0-9>+
<ereal>             ::= <0-9>*"."<0-9>+<eE><sign><0-9>+
<eE>                ::= "e" | "E"
<symbol>            ::= <a-zA-Z><a-zA-Z0-9>*
<variable>          ::= <symbol> | <obj_ident>
<inp_obj_symb>      ::= <symbol>
                        /* will be replaced by an actual
                           object name of an active object from input */

<out_obj_symb>      ::= <symbol>
                        /* will be replaced by an actual
                           object name of an active object from the
                           representation of nuclide data */

<obj_class>         ::= "PARTICLE" | "NUCLIDE"
<obj_attr>          ::= <nuclide_attr> | <particle_attr>
<obj_ident>         ::= <nuclide_name>"."<nuclide_attr> |
                           <particle_name>"."<particle_attr>
<out_obj_ident>     ::= <out_obj_symb>"."<obj_attr>

<input-file>        ::= <simulation_time><report_time><cycle_time>
                        <report_var>*<debug_var>*<history_var>*
                        <nuclide_struct>*<particle_struct>*
                        <equip_struct>*

<simulation_time>   ::= "simulation_time" <time>
<report_time>       ::= "report_time" <time>
<cycle_time         ::= "cycle_time" <time>
<time>              ::= <number> <unit_time>
<unit_time>         ::= "a" | "d" | "h" | "m" | "s"
<report_va          ::= "report" <variable>
<debug_var>         ::= "debug" <variable>
<history_var        ::= "history" <variable>

<nuclide_struct>    ::= "NUCLIDE" <nuclide_name>
                        "{" <nuclide_spec>+ "}"
<nuclide_name>      ::= <A-Z><a-z>*<0-9>+
<nuclide_spec>      ::= <nuclide_attr><number>
```

```
<nuclide_attr>      ::= "Z" | "N" | "A" | "amount" |
                        "energy" | "sigma_a" | "sigma_f"|
                        "density" | "sigma_g"

<particle_struct>   ::= "PARTICLE" <particle_name>
                        "{" <particle_spec>+ "}"
<particle_name>     ::= <A-Z><a-z>*
<particle_spec>     ::= <particle_attr><number>
<particle_attr>     ::= "Z" | "A" | "amount" | "energy" | "id"

<equip_struct>      ::= <equip_name> <equip_symb>
                        "{"<equip_spec>+"}"
<equip_name>        ::= <symbol>
<equip_symb>        ::= <symbol>
<equip_spec>        ::= <symbol> <number>

<process_defs>      ::= <process_def>+
<process_def>       ::= <basic_def> <aggre_def>+

<basic_def>         ::= <basic_heading> "{" <basic_body> "}"

<basic_heading>     ::= "basic_nuclear_physics_model"
                        <basic_id> "is" <process_name>

<basic_body>        ::= <reactants> <act_conds> <interm_state>**
                        <product_basic>

<process_id>        ::= <symbol>
<process_name>      ::= <symbol>

<reactants>         ::= "reactants" "{" <reactant_spec>+ "}"
<reactant_spec>     ::= <inp_obj_symb> "is" <obj_class>
<act_conds>         ::= "activity_conditions" "{" <bool_expr> "}"
<bool_exp>          ::= <operand> <bool_operator> <operand>
<operand>           ::= <number> | <obj_ident> | <bool_expr>
<act_operator>      ::= <boolean_op> | <relational_op>
<relational_op>     ::= "=" | "!=" | "<" | ">" | "<=" | ">="
<boolean_op>        ::= "&" | "|" | "!"
<weak_op>           ::= "+" | "-"
<strong_op>         ::= "*" | "/"

<interm_state>      ::= "intermediate_state" "{" <NPT_func>
```

```
                        <symbol> "(" <inp_obj_symb> ")" "}"
<NPT_func>          ::= <symbol>"_"<a-zA-Z0-9>*
                        /* "neutron_absorption" */

<product_basic>     ::= "products" "{" <object_def>+ "}"

<object_def>        ::= "def_obj" "{" <out_obj_symb> <obj_class>
                                      <obj_spec> "}"

<obj_spec>          ::= <det_spec> | <prob_spec>
<det_spec>          ::= <out_obj_symb>"."<obj_attr> "is"
                        "DETERMINISTIC" "(" "value" ":" <expr> ")"

<expr>              ::= <term> | <expr> <weak_op> <term>
<term>              ::= <element> | <term> <strong_op><element>
<element>           ::= <number> | <variable> | "(" <expr> ")"

<prob_spec>         ::= <out_obj_symb>"."<obj_attr> "is"
                           "PROBABILISTIC" "(" "value" ":"
                           <decomp_lookup> ")"
<decomp_lookup>     ::= <look_up_tab_vals> | <decomp_func>

<look_up_tab_vals> ::= "look_up_table" "(" <vector_var> <pdf_file>
                        "("<lb>","<ub>")" ")"
<vector_var>        ::= <symbol>
<vector_elm>        ::= <vector_var>"["<symbol>"]" |
                        <vector_var> "(" <obj_symb> ")" "["<symbol>"]"

<pdf_file>          ::= <symbol>
                        /*  file name containning pdf values
                        e.g. "neutron_sp" | "gamma_sp" |
                        "mass_yieldA1"| "mass_yieldA2" |
                        "energy_distrF1" | "energy_distrF2" */

<decomp_func>       ::= "decomposition" "(" <pdf_var><distr_func> ")"
<distr_func>        ::= <distr_name> "(" <mean>","<variance>",
                                      " <lb>","<ub>" ")"
<distr_name>        ::= "Gaussian" | "Poisson" | "Uniform" | "Xsquare"
<mean>              ::= <number>
<variance>          ::= <number>
<lb>                ::= <integer>
<ub>                ::= <integer>
```

```
<aggre_def>          ::= <aggre_heading> "{" <aggre_body> "}"
<aggre_heading>      ::= "aggregate_effects_model"
                          <process_name>  "(" <basics_id> ")"

<aggre_body>         ::= <material><equipment><preconds>
                         <act_conds><param_reacts><relations>
                         <product_aggr>

<material>           ::= <inp_obj_symb>
<equipment>          ::= "equipment" "{" <equip_spec>+  "}"
<equip_spec>         ::= <equip_name> "is" <equip_clas>
<equip_class>        ::= <A-Z>+

<pre_conds>          ::= "pre_conditions" "{" <equip_sta> "}"
<equip_sta>          ::=  <equip_ident> "is" <status>
<status>             ::= "ON" | "OFF" | "0" | "1"

<param_reacts>       ::= "parametric_reactions" "{" <param_def>* "}
<param_def>          ::= <def_const>*<def_vector>*
<def_const>          ::= "def_const" <symbol> "(" "value" ":"
                         <number> ")"
<def_vector>         ::= "def_vector" <symbol> "(" "value" ":"
                         "look_up_table" "(" <pdf_file>
                                          "("<lb>","<ub>")" ")"

<relations>          ::= "relations" "{" <infl_def>* "}"
<infl_def>           ::= <alg_infl>* <cum_infl>* <aggr_infl>*
                         <distr_infl>* <decay_infl>*

<alg_infl>           ::= "alg_infl" "{" "(" <infl_ed_var> ")"
                         "(" <list_infl_ing_vars> ")"
                         "(" <funct_spec> ")"  "}"

<infl_ed_var>        ::= <npt_symb_var>
<npt_symb_var>       ::= <symbol> | <inp_symb_var> | <symb_var>
<inp_symb_var>       ::= <inp_obj_symb>"."<obj_attr>
<symb_var>           ::= <symbol>"("<inp_obj_symb>")"

<list_infl_ing_vars>::= <npt_symb_var> |
                         <npt_symb_var>","<list_infl_ing_vars>
```

```
<funct_spec>        ::= <infl_expr> | "sum" "(" <vect_expr> ")"

<infl_expr>         ::= <infl_term> | <infl_expr> <weak_op>
                        <infl_term>
<infl_term>         ::= <infl_element> | <infl_term> <strong_op>
                        <infl_element>
<inlf_element>      ::= <number> | <npt_symb_var> |
                        "(" <infl_expr> ")"

<vect_expr>         ::= <in_vect_expr> "," <loop_counter>

<loop_counter>      ::= <symbol> "=" <lb> ".." <ub>

<in_vect_expr>      ::= <in_vect_term> |
                        <in_vect_expr> <weak_op> <in_vect_term>
<in_vect_term>      ::= <in_vect_element> | <in_vect_term> <strong_op>
                        <in_vect_element>
<in_vect_element>   ::= <number> | <in_vect_var> |
                        "(" <in_vect_expr> ")"

<in_vect_var>       ::= <npt_symb_var> | <vector_elm>

<aggr_infl>         ::= "aggr_infl" "{" "(" <infl_ed_vect_var> ")"
                        "(" <infl_ing_vects_vars_vects>  ")"
                        "(" <func_spec> ")" "}"

<infl_ing_vects_vars_vects> ::= <infl_ing_vect_vars> |
                          <infl_ing_var_and_vect_vars>
<infl_ed_vect_var>  ::= <vector_elm>
<infl_ing_vect_vars>::= <vector_var>
<infl_ing_var_vect_vars> ::= [<vector_var>] |
                          <vector_var> "," <npt_symb_var> |
                          <npt_symb_var> "," <vector_var> |
                          <vector_var><infl_ing_var_vect_vars> |
                          <npt_symb_var> "," <infl_ing_var_vect_vars>

<funct spec>         ::= <vect_expr>

<cum_infl>           ::= "cum_infl" "{" "(" <infl_ed_var> ")"
                        "(" <list_infl_ing_vars> ")"
                        "(" <funct_spec> ")"  "}"
```

```
<distr_infl>        ::= "distr_infl" "{" "(" <out_obj_symb> ")"
                        "(" <infl_ing_vects_vars_vects>  ")"
                        "(" <loop_expr> ")" "}"
<loop_expr>         ::= <in_loop_expr>+  <loop_counter> <more_loop>*
<more_loop>         ::= "," <loop_counter>
<in_loop_expr>      ::= <out_obj_ident> "=" <func_spec> ";"

<decay_infl>        ::= "decay_infl" "{" "(" <cond_expr> ")"
                        "(" <infl_ed_var> ")" "(" <infl_ing_vars> ")"
                         <infl_expr> "}"

<product_aggr>      ::= "products" "{" <cum_infl>* <distr_infl>* "}"
                        <vector_var> |
                        <vector_var> "," <list_infl_ing_vars>
```

Appendix IV

Set of Active Processes at the End of the Second Cycle

This appendix shows the different types of nuclides that are available in the system at the end of the second cycle of running the simulation for Experiment 2.

```
U235
neutron
PHWNUR
HW
Zra
BNPM radioactive_capture(U235,neutron)
AEM (BNPM radioactive_capture(U235,neutron))
BNPM fission(U235,neutron)
AEM (BNPM fission(U235,neutron),PHWNUR,HW,Zra)

U238
BNPM radioactive_capture(U238,neutron)
AEM (BNPM radioactive_capture(U238,neutron))

U236
BNPM radioactive_capture(U236,neutron)
AEM (BNPM radioactive_capture(U236,neutron))

U239
BNPM beta_negative_decay(U239)
AEM (BNPM beta_negative_decay(U239))

Tb158
BNPM beta_negative_decay(Tb158)
AEM (BNPM beta_negative_decay(Tb158))

Tb159
BNPM radioactive_capture(Tb159,neutron)
AEM (BNPM radioactive_capture(Tb159,neutron))

Gd155
BNPM radioactive_capture(Gd155,neutron)
AEM (BNPM radioactive_capture(Gd155,neutron))

Gd156
BNPM radioactive_capture(Gd156,neutron)
AEM (BNPM radioactive_capture(Gd156,neutron))

Gd157
BNPM radioactive_capture(Gd157,neutron)
AEM (BNPM radioactive_capture(Gd157,neutron))
```

Gd158

Gd159
BNPM beta_negative_decay(Gd159)
AEM (BNPM beta_negative_decay(Gd159))

Eu153
BNPM radioactive_capture(Eu153,neutron)
AEM (BNPM radioactive_capture(Eu153,neutron))

Eu154
BNPM radioactive_capture(Eu154,neutron)
AEM (BNPM radioactive_capture(Eu154,neutron))
BNPM beta_positive_decay(Eu154)
AEM (BNPM beta_positive_decay(Eu154))

Eu155
BNPM radioactive_capture(Eu155,neutron)
AEM (BNPM radioactive_capture(Eu155,neutron))
BNPM beta_negative_decay(Eu155)
AEM (BNPM beta_negative_decay(Eu155))

Eu156
BNPM beta_negative_decay(Eu156)
AEM (BNPM beta_negative_decay(Eu156))

Eu157
BNPM beta_negative_decay(Eu157)
AEM (BNPM beta_negative_decay(Eu157))

Eu158
BNPM beta_negative_decay(Eu158)
AEM (BNPM beta_negative_decay(Eu158))

Eu159
BNPM beta_negative_decay(Eu159)
AEM (BNPM beta_negative_decay(Eu159))

Sm150
BNPM radioactive_capture(Sm150,neutron)
AEM (BNPM radioactive_capture(Sm150,neutron))

```
Sm151
BNPM radioactive_capture(Sm151,neutron)
AEM (BNPM radioactive_capture(Sm151,neutron))
BNPM beta_negative_decay(Sm151)
AEM (BNPM beta_negative_decay(Sm151))

Sm152
BNPM radioactive_capture(Sm152,neutron)
AEM (BNPM radioactive_capture(Sm152,neutron))

Sm153
BNPM radioactive_capture(Sm153,neutron)
AEM (BNPM radioactive_capture(Sm153,neutron))
BNPM beta_negative_decay(Sm153)
AEM (BNPM beta_negative_decay(Sm153))

Sm154
BNPM radioactive_capture(Sm154,neutron)
AEM (BNPM radioactive_capture(Sm154,neutron))

Sm155
BNPM beta_negative_decay(Sm155)
AEM (BNPM beta_negative_decay(Sm155))

Sm156
BNPM beta_negative_decay(Sm156)
AEM (BNPM beta_negative_decay(Sm156))

Sm157
BNPM beta_negative_decay(Sm157)
AEM (BNPM beta_negative_decay(Sm157))

Sm158
BNPM beta_negative_decay(Sm158)
AEM (BNPM beta_negative_decay(Sm158))

Sm159
BNPM beta_negative_decay(Sm159)
AEM (BNPM beta_negative_decay(Sm159))

Pm148
BNPM radioactive_capture(Pm148,neutron)
```

```
AEM (BNPM radioactive_capture(Pm148,neutron))
BNPM beta_negative_decay(Pm148)
AEM (BNPM beta_negative_decay(Pm148))

Pm149
BNPM beta_negative_decay(Pm149)
AEM (BNPM beta_negative_decay(Pm149))

Pm150
BNPM beta_negative_decay(Pm150)
AEM (BNPM beta_negative_decay(Pm150))

Pm151
BNPM beta_negative_decay(Pm151)
AEM (BNPM beta_negative_decay(Pm151))

Pm152
BNPM beta_negative_decay(Pm152)
AEM (BNPM beta_negative_decay(Pm152))

Pm153
BNPM beta_negative_decay(Pm153)
AEM (BNPM beta_negative_decay(Pm153))

Pm154
BNPM beta_negative_decay(Pm154)
AEM (BNPM beta_negative_decay(Pm154))

Pm155
BNPM beta_negative_decay(Pm155)
AEM (BNPM beta_negative_decay(Pm155))

Pm156
BNPM beta_negative_decay(Pm156)
AEM (BNPM beta_negative_decay(Pm156))

Pm157
BNPM beta_negative_decay(Pm157)
AEM (BNPM beta_negative_decay(Pm157))

Pm158
BNPM beta_negative_decay(Pm158)
```

```
AEM (BNPM beta_negative_decay(Pm158))

Nd145
BNPM radioactive_capture(Nd145,neutron)
AEM (BNPM radioactive_capture(Nd145,neutron))

Nd146
BNPM radioactive_capture(Nd146,neutron)
AEM (BNPM radioactive_capture(Nd146,neutron))

Nd147
BNPM beta_negative_decay(Nd147)
AEM (BNPM beta_negative_decay(Nd147))

Nd148
BNPM radioactive_capture(Nd148,neutron)
AEM (BNPM radioactive_capture(Nd148,neutron))

Nd149
BNPM beta_negative_decay(Nd149)
AEM (BNPM beta_negative_decay(Nd149))

Nd150
BNPM radioactive_capture(Nd150,neutron)
AEM (BNPM radioactive_capture(Nd150,neutron))

Nd151
BNPM beta_negative_decay(Nd151)
AEM (BNPM beta_negative_decay(Nd151))

Nd152
BNPM beta_negative_decay(Nd152)
AEM (BNPM beta_negative_decay(Nd152))

Nd153
BNPM beta_negative_decay(Nd153)
AEM (BNPM beta_negative_decay(Nd153))

Nd154
BNPM beta_negative_decay(Nd154)
AEM (BNPM beta_negative_decay(Nd154))
```

```
Nd155
BNPM beta_negative_decay(Nd155)
AEM (BNPM beta_negative_decay(Nd155))

Nd156
BNPM beta_negative_decay(Nd156)
AEM (BNPM beta_negative_decay(Nd156))

Pr143
BNPM radioactive_capture(Pr143,neutron)
AEM (BNPM radioactive_capture(Pr143,neutron))
BNPM beta_negative_decay(Pr143)
AEM (BNPM beta_negative_decay(Pr143))

Pr144
BNPM beta_negative_decay(Pr144)
AEM (BNPM beta_negative_decay(Pr144))

Pr145
BNPM beta_negative_decay(Pr145)
AEM (BNPM beta_negative_decay(Pr145))

Pr146
BNPM beta_negative_decay(Pr146)
AEM (BNPM beta_negative_decay(Pr146))

Pr147
BNPM beta_negative_decay(Pr147)
AEM (BNPM beta_negative_decay(Pr147))

Pr148
BNPM beta_negative_decay(Pr148)
AEM (BNPM beta_negative_decay(Pr148))

Pr149
BNPM beta_negative_decay(Pr149)
AEM (BNPM beta_negative_decay(Pr149))

Pr150
BNPM beta_negative_decay(Pr150)
AEM (BNPM beta_negative_decay(Pr150))
```

```
Pr151
BNPM beta_negative_decay(Pr151)
AEM (BNPM beta_negative_decay(Pr151))

Pr152
BNPM beta_negative_decay(Pr152)
AEM (BNPM beta_negative_decay(Pr152))

Pr153
BNPM beta_negative_decay(Pr153)
AEM (BNPM beta_negative_decay(Pr153))

Pr154
BNPM beta_negative_decay(Pr154)
AEM (BNPM beta_negative_decay(Pr154))

Ce140

Ce141
BNPM beta_negative_decay(Ce141)
AEM (BNPM beta_negative_decay(Ce141))

Ce142

Ce143
BNPM beta_negative_decay(Ce143)
AEM (BNPM beta_negative_decay(Ce143))

Ce144
BNPM beta_negative_decay(Ce144)
AEM (BNPM beta_negative_decay(Ce144))

Ce145
BNPM beta_negative_decay(Ce145)
AEM (BNPM beta_negative_decay(Ce145))

Ce146
BNPM beta_negative_decay(Ce146)
AEM (BNPM beta_negative_decay(Ce146))

Ce147
BNPM beta_negative_decay(Ce147)
```

```
AEM (BNPM beta_negative_decay(Ce147))

Ce148
BNPM beta_negative_decay(Ce148)
AEM (BNPM beta_negative_decay(Ce148))

Ce149
BNPM beta_negative_decay(Ce149)
AEM (BNPM beta_negative_decay(Ce149))

Ce150
BNPM beta_negative_decay(Ce150)
AEM (BNPM beta_negative_decay(Ce150))

Ce151
BNPM beta_negative_decay(Ce151)
AEM (BNPM beta_negative_decay(Ce151))

Ce152
BNPM beta_negative_decay(Ce152)
AEM (BNPM beta_negative_decay(Ce152))

La138
BNPM radioactive_capture(La138,neutron)
AEM (BNPM radioactive_capture(La138,neutron))

La139

La140
BNPM beta_negative_decay(La140)
AEM (BNPM beta_negative_decay(La140))

La141
BNPM beta_negative_decay(La141)
AEM (BNPM beta_negative_decay(La141))

La142
BNPM beta_negative_decay(La142)
AEM (BNPM beta_negative_decay(La142))

La143
BNPM beta_negative_decay(La143)
```

```
AEM (BNPM beta_negative_decay(La143))

La144
BNPM beta_negative_decay(La144)
AEM (BNPM beta_negative_decay(La144))

La145
BNPM beta_negative_decay(La145)
AEM (BNPM beta_negative_decay(La145))

La146
BNPM beta_negative_decay(La146)
AEM (BNPM beta_negative_decay(La146))

La147
BNPM beta_negative_decay(La147)
AEM (BNPM beta_negative_decay(La147))

La148
BNPM beta_negative_decay(La148)
AEM (BNPM beta_negative_decay(La148))

La149
BNPM beta_negative_decay(La149)
AEM (BNPM beta_negative_decay(La149))

Ba135

Ba136

Ba137

Ba138

Ba139
BNPM beta_negative_decay(Ba139)
AEM (BNPM beta_negative_decay(Ba139))

Ba140
BNPM beta_negative_decay(Ba140)
AEM (BNPM beta_negative_decay(Ba140))
```

```
Ba141
BNPM beta_negative_decay(Ba141)
AEM (BNPM beta_negative_decay(Ba141))

Ba142
BNPM beta_negative_decay(Ba142)
AEM (BNPM beta_negative_decay(Ba142))

Ba143
BNPM beta_negative_decay(Ba143)
AEM (BNPM beta_negative_decay(Ba143))

Ba144
BNPM beta_negative_decay(Ba144)
AEM (BNPM beta_negative_decay(Ba144))

Ba145
BNPM beta_negative_decay(Ba145)
AEM (BNPM beta_negative_decay(Ba145))

Ba146
BNPM beta_negative_decay(Ba146)
AEM (BNPM beta_negative_decay(Ba146))

Ba147
BNPM beta_negative_decay(Ba147)
AEM (BNPM beta_negative_decay(Ba147))

Ba148
BNPM beta_negative_decay(Ba148)
AEM (BNPM beta_negative_decay(Ba148))

Ba149
BNPM beta_negative_decay(Ba149)
AEM (BNPM beta_negative_decay(Ba149))

Cs133
BNPM radioactive_capture(Cs133,neutron)
AEM (BNPM radioactive_capture(Cs133,neutron))

Cs134
BNPM beta_negative_decay(Cs134)
```

```
AEM (BNPM beta_negative_decay(Cs134))

Cs135

Cs136
BNPM beta_negative_decay(Cs136)
AEM (BNPM beta_negative_decay(Cs136))

Cs137
BNPM beta_negative_decay(Cs137)
AEM (BNPM beta_negative_decay(Cs137))

Cs138
BNPM beta_negative_decay(Cs138)
AEM (BNPM beta_negative_decay(Cs138))

Cs139
BNPM beta_negative_decay(Cs139)
AEM (BNPM beta_negative_decay(Cs139))

Cs140
BNPM beta_negative_decay(Cs140)
AEM (BNPM beta_negative_decay(Cs140))

Cs141
BNPM beta_negative_decay(Cs141)
AEM (BNPM beta_negative_decay(Cs141))

Cs142
BNPM beta_negative_decay(Cs142)
AEM (BNPM beta_negative_decay(Cs142))

Cs143
BNPM beta_negative_decay(Cs143)
AEM (BNPM beta_negative_decay(Cs143))

Cs144
BNPM beta_negative_decay(Cs144)
AEM (BNPM beta_negative_decay(Cs144))

Cs145
BNPM beta_negative_decay(Cs145)
```

```
AEM (BNPM beta_negative_decay(Cs145))

Cs146
BNPM beta_negative_decay(Cs146)
AEM (BNPM beta_negative_decay(Cs146))

Cs147
BNPM beta_negative_decay(Cs147)
AEM (BNPM beta_negative_decay(Cs147))

Cs148
BNPM beta_negative_decay(Cs148)
AEM (BNPM beta_negative_decay(Cs148))

Xe130
BNPM radioactive_capture(Xe130,neutron)
AEM (BNPM radioactive_capture(Xe130,neutron))

Xe131
BNPM radioactive_capture(Xe131,neutron)
AEM (BNPM radioactive_capture(Xe131,neutron))

Xe132

Xe133
BNPM beta_negative_decay(Xe133)
AEM (BNPM beta_negative_decay(Xe133))

Xe134

Xe135
BNPM radioactive_capture(Xe135,neutron)
AEM (BNPM radioactive_capture(Xe135,neutron))
BNPM beta_negative_decay(Xe135)
AEM (BNPM beta_negative_decay(Xe135))

Xe136

Xe137
BNPM beta_negative_decay(Xe137)
AEM (BNPM beta_negative_decay(Xe137))
```

```
Xe138
BNPM beta_negative_decay(Xe138)
AEM (BNPM beta_negative_decay(Xe138))

Xe139
BNPM beta_negative_decay(Xe139)
AEM (BNPM beta_negative_decay(Xe139))

Xe140
BNPM beta_negative_decay(Xe140)
AEM (BNPM beta_negative_decay(Xe140))

Xe141
BNPM beta_negative_decay(Xe141)
AEM (BNPM beta_negative_decay(Xe141))

Xe142
BNPM beta_negative_decay(Xe142)
AEM (BNPM beta_negative_decay(Xe142))

Xe143
BNPM beta_negative_decay(Xe143)
AEM (BNPM beta_negative_decay(Xe143))

Xe144
BNPM beta_negative_decay(Xe144)
AEM (BNPM beta_negative_decay(Xe144))

Xe145
BNPM beta_negative_decay(Xe145)
AEM (BNPM beta_negative_decay(Xe145))

I127
BNPM radioactive_capture(I127,neutron)
AEM (BNPM radioactive_capture(I127,neutron))

I128
BNPM beta_negative_decay(I128)
AEM (BNPM beta_negative_decay(I128))

I129
BNPM radioactive_capture(I129,neutron)
```

```
AEM (BNPM radioactive_capture(I129,neutron))

I130
BNPM beta_negative_decay(I130)
AEM (BNPM beta_negative_decay(I130))

I131
BNPM beta_negative_decay(I131)
AEM (BNPM beta_negative_decay(I131))

I132
BNPM beta_negative_decay(I132)
AEM (BNPM beta_negative_decay(I132))

I133
BNPM beta_negative_decay(I133)
AEM (BNPM beta_negative_decay(I133))

I134
BNPM beta_negative_decay(I134)
AEM (BNPM beta_negative_decay(I134))

I135
BNPM beta_negative_decay(I135)
AEM (BNPM beta_negative_decay(I135))

I136
BNPM beta_negative_decay(I136)
AEM (BNPM beta_negative_decay(I136))

I137
BNPM beta_negative_decay(I137)
AEM (BNPM beta_negative_decay(I137))

I138
BNPM beta_negative_decay(I138)
AEM (BNPM beta_negative_decay(I138))

I139
BNPM beta_negative_decay(I139)
AEM (BNPM beta_negative_decay(I139))
```

```
I140
BNPM beta_negative_decay(I140)
AEM (BNPM beta_negative_decay(I140))

I141
BNPM beta_negative_decay(I141)
AEM (BNPM beta_negative_decay(I141))

I142
BNPM beta_negative_decay(I142)
AEM (BNPM beta_negative_decay(I142))

Te125

Te126

Te127
BNPM beta_negative_decay(Te127)
AEM (BNPM beta_negative_decay(Te127))

Te128

Te129
BNPM beta_negative_decay(Te129)
AEM (BNPM beta_negative_decay(Te129))

Te130

Te131
BNPM beta_negative_decay(Te131)
AEM (BNPM beta_negative_decay(Te131))

Te132
BNPM beta_negative_decay(Te132)
AEM (BNPM beta_negative_decay(Te132))

Te133
BNPM beta_negative_decay(Te133)
AEM (BNPM beta_negative_decay(Te133))

Te134
BNPM beta_negative_decay(Te134)
```

```
AEM (BNPM beta_negative_decay(Te134))

Te135
BNPM beta_negative_decay(Te135)
AEM (BNPM beta_negative_decay(Te135))

Te136
BNPM beta_negative_decay(Te136)
AEM (BNPM beta_negative_decay(Te136))

Te137
BNPM beta_negative_decay(Te137)
AEM (BNPM beta_negative_decay(Te137))

Te138
BNPM beta_negative_decay(Te138)
AEM (BNPM beta_negative_decay(Te138))

Sb122
BNPM beta_negative_decay(Sb122)
AEM (BNPM beta_negative_decay(Sb122))

Sb123
BNPM radioactive_capture(Sb123,neutron)
AEM (BNPM radioactive_capture(Sb123,neutron))

Sb124
BNPM beta_negative_decay(Sb124)
AEM (BNPM beta_negative_decay(Sb124))

Sb125
BNPM beta_negative_decay(Sb125)
AEM (BNPM beta_negative_decay(Sb125))

Sb126
BNPM beta_negative_decay(Sb126)
AEM (BNPM beta_negative_decay(Sb126))

Sb127
BNPM beta_negative_decay(Sb127)
AEM (BNPM beta_negative_decay(Sb127))
```

```
Sb128
BNPM beta_negative_decay(Sb128)
AEM (BNPM beta_negative_decay(Sb128))

Sb129
BNPM beta_negative_decay(Sb129)
AEM (BNPM beta_negative_decay(Sb129))

Sb130
BNPM beta_negative_decay(Sb130)
AEM (BNPM beta_negative_decay(Sb130))

Sb131
BNPM beta_negative_decay(Sb131)
AEM (BNPM beta_negative_decay(Sb131))

Sb132
BNPM beta_negative_decay(Sb132)
AEM (BNPM beta_negative_decay(Sb132))

Sb133
BNPM beta_negative_decay(Sb133)
AEM (BNPM beta_negative_decay(Sb133))

Sb134
BNPM beta_negative_decay(Sb134)
AEM (BNPM beta_negative_decay(Sb134))

Sb135
BNPM beta_negative_decay(Sb135)
AEM (BNPM beta_negative_decay(Sb135))

Sb136
BNPM beta_negative_decay(Sb136)
AEM (BNPM beta_negative_decay(Sb136))

Sn120

Sn121
BNPM beta_negative_decay(Sn121)
AEM (BNPM beta_negative_decay(Sn121))
```

```
Sn122

Sn123
BNPM beta_negative_decay(Sn123)
AEM (BNPM beta_negative_decay(Sn123))

Sn124

Sn125
BNPM beta_negative_decay(Sn125)
AEM (BNPM beta_negative_decay(Sn125))

Sn126

Sn127
BNPM beta_negative_decay(Sn127)
AEM (BNPM beta_negative_decay(Sn127))

Sn128
BNPM beta_negative_decay(Sn128)
AEM (BNPM beta_negative_decay(Sn128))

Sn129
BNPM beta_negative_decay(Sn129)
AEM (BNPM beta_negative_decay(Sn129))

Sn130
BNPM beta_negative_decay(Sn130)
AEM (BNPM beta_negative_decay(Sn130))

Sn131
BNPM beta_negative_decay(Sn131)
AEM (BNPM beta_negative_decay(Sn131))

Sn132
BNPM beta_negative_decay(Sn132)
AEM (BNPM beta_negative_decay(Sn132))

Sn133
BNPM beta_negative_decay(Sn133)
AEM (BNPM beta_negative_decay(Sn133))
```

```
Sn134
BNPM beta_negative_decay(Sn134)
AEM (BNPM beta_negative_decay(Sn134))

In117
BNPM beta_negative_decay(In117)
AEM (BNPM beta_negative_decay(In117))

In118
BNPM beta_negative_decay(In118)
AEM (BNPM beta_negative_decay(In118))

In119
BNPM beta_negative_decay(In119)
AEM (BNPM beta_negative_decay(In119))

In120
BNPM beta_negative_decay(In120)
AEM (BNPM beta_negative_decay(In120))

In121
BNPM beta_negative_decay(In121)
AEM (BNPM beta_negative_decay(In121))

In122
BNPM beta_negative_decay(In122)
AEM (BNPM beta_negative_decay(In122))

In123
BNPM beta_negative_decay(In123)
AEM (BNPM beta_negative_decay(In123))

In124
BNPM beta_negative_decay(In124)
AEM (BNPM beta_negative_decay(In124))

In125
BNPM beta_negative_decay(In125)
AEM (BNPM beta_negative_decay(In125))

In126
BNPM beta_negative_decay(In126)
```

```
AEM (BNPM beta_negative_decay(In126))

In127
BNPM beta_negative_decay(In127)
AEM (BNPM beta_negative_decay(In127))

In128
BNPM beta_negative_decay(In128)
AEM (BNPM beta_negative_decay(In128))

In129
BNPM beta_negative_decay(In129)
AEM (BNPM beta_negative_decay(In129))

In130
BNPM beta_negative_decay(In130)
AEM (BNPM beta_negative_decay(In130))

In131
BNPM beta_negative_decay(In131)
AEM (BNPM beta_negative_decay(In131))

In132
BNPM beta_negative_decay(In132)
AEM (BNPM beta_negative_decay(In132))

In133
BNPM beta_negative_decay(In133)
AEM (BNPM beta_negative_decay(In133))

Cd115
BNPM beta_negative_decay(Cd115)
AEM (BNPM beta_negative_decay(Cd115))

Cd116

Cd117
BNPM beta_negative_decay(Cd117)
AEM (BNPM beta_negative_decay(Cd117))

Cd118
BNPM beta_negative_decay(Cd118)
```

```
AEM (BNPM beta_negative_decay(Cd118))

Cd119
BNPM beta_negative_decay(Cd119)
AEM (BNPM beta_negative_decay(Cd119))

Cd120
BNPM beta_negative_decay(Cd120)
AEM (BNPM beta_negative_decay(Cd120))

Cd121
BNPM beta_negative_decay(Cd121)
AEM (BNPM beta_negative_decay(Cd121))

Cd122
BNPM beta_negative_decay(Cd122)
AEM (BNPM beta_negative_decay(Cd122))

Cd123
BNPM beta_negative_decay(Cd123)
AEM (BNPM beta_negative_decay(Cd123))

Cd124
BNPM beta_negative_decay(Cd124)
AEM (BNPM beta_negative_decay(Cd124))

Cd125
BNPM beta_negative_decay(Cd125)
AEM (BNPM beta_negative_decay(Cd125))

Cd126
BNPM beta_negative_decay(Cd126)
AEM (BNPM beta_negative_decay(Cd126))

Cd127
BNPM beta_negative_decay(Cd127)
AEM (BNPM beta_negative_decay(Cd127))

Cd128
BNPM beta_negative_decay(Cd128)
AEM (BNPM beta_negative_decay(Cd128))
```

```
Cd129
BNPM beta_negative_decay(Cd129)
AEM (BNPM beta_negative_decay(Cd129))

Cd130
BNPM beta_negative_decay(Cd130)
AEM (BNPM beta_negative_decay(Cd130))

Ag112
BNPM beta_negative_decay(Ag112)
AEM (BNPM beta_negative_decay(Ag112))

Ag113
BNPM beta_negative_decay(Ag113)
AEM (BNPM beta_negative_decay(Ag113))

Ag114
BNPM beta_negative_decay(Ag114)
AEM (BNPM beta_negative_decay(Ag114))

Ag115
BNPM beta_negative_decay(Ag115)
AEM (BNPM beta_negative_decay(Ag115))

Ag116
BNPM beta_negative_decay(Ag116)
AEM (BNPM beta_negative_decay(Ag116))

Ag117
BNPM beta_negative_decay(Ag117)
AEM (BNPM beta_negative_decay(Ag117))

Ag118
BNPM beta_negative_decay(Ag118)
AEM (BNPM beta_negative_decay(Ag118))

Ag119
BNPM beta_negative_decay(Ag119)
AEM (BNPM beta_negative_decay(Ag119))

Ag120
BNPM beta_negative_decay(Ag120)
```

```
AEM (BNPM beta_negative_decay(Ag120))

Ag121
BNPM beta_negative_decay(Ag121)
AEM (BNPM beta_negative_decay(Ag121))

Ag122
BNPM beta_negative_decay(Ag122)
AEM (BNPM beta_negative_decay(Ag122))

Ag123
BNPM beta_negative_decay(Ag123)
AEM (BNPM beta_negative_decay(Ag123))

Ag124
BNPM beta_negative_decay(Ag124)
AEM (BNPM beta_negative_decay(Ag124))

Pd110

Pd111
BNPM beta_negative_decay(Pd111)
AEM (BNPM beta_negative_decay(Pd111))

Pd112
BNPM beta_negative_decay(Pd112)
AEM (BNPM beta_negative_decay(Pd112))

Pd113
BNPM beta_negative_decay(Pd113)
AEM (BNPM beta_negative_decay(Pd113))

Pd114
BNPM beta_negative_decay(Pd114)
AEM (BNPM beta_negative_decay(Pd114))

Pd115
BNPM beta_negative_decay(Pd115)
AEM (BNPM beta_negative_decay(Pd115))

Pd116
BNPM beta_negative_decay(Pd116)
```

```
AEM (BNPM beta_negative_decay(Pd116))

Pd117
BNPM beta_negative_decay(Pd117)
AEM (BNPM beta_negative_decay(Pd117))

Pd118
BNPM beta_negative_decay(Pd118)
AEM (BNPM beta_negative_decay(Pd118))

Rh107
BNPM beta_negative_decay(Rh107)
AEM (BNPM beta_negative_decay(Rh107))

Rh108
BNPM beta_negative_decay(Rh108)
AEM (BNPM beta_negative_decay(Rh108))

Rh109
BNPM beta_negative_decay(Rh109)
AEM (BNPM beta_negative_decay(Rh109))

Rh110
BNPM beta_negative_decay(Rh110)
AEM (BNPM beta_negative_decay(Rh110))

Rh111
BNPM beta_negative_decay(Rh111)
AEM (BNPM beta_negative_decay(Rh111))

Rh112
BNPM beta_negative_decay(Rh112)
AEM (BNPM beta_negative_decay(Rh112))

Rh113
BNPM beta_negative_decay(Rh113)
AEM (BNPM beta_negative_decay(Rh113))

Rh114
BNPM beta_negative_decay(Rh114)
AEM (BNPM beta_negative_decay(Rh114))
```

```
Rh115
BNPM beta_negative_decay(Rh115)
AEM (BNPM beta_negative_decay(Rh115))

Rh116
BNPM beta_negative_decay(Rh116)
AEM (BNPM beta_negative_decay(Rh116))

Ru105
BNPM beta_negative_decay(Ru105)
AEM (BNPM beta_negative_decay(Ru105))

Ru106
BNPM beta_negative_decay(Ru106)
AEM (BNPM beta_negative_decay(Ru106))

Ru107
BNPM beta_negative_decay(Ru107)
AEM (BNPM beta_negative_decay(Ru107))

Ru108
BNPM beta_negative_decay(Ru108)
AEM (BNPM beta_negative_decay(Ru108))

Ru109
BNPM beta_negative_decay(Ru109)
AEM (BNPM beta_negative_decay(Ru109))

Ru110
BNPM beta_negative_decay(Ru110)
AEM (BNPM beta_negative_decay(Ru110))

Ru111
BNPM beta_negative_decay(Ru111)
AEM (BNPM beta_negative_decay(Ru111))

Ru112
BNPM beta_negative_decay(Ru112)
AEM (BNPM beta_negative_decay(Ru112))

Ru113
BNPM beta_negative_decay(Ru113)
```

```
AEM (BNPM beta_negative_decay(Ru113))

Tc102
BNPM beta_negative_decay(Tc102)
AEM (BNPM beta_negative_decay(Tc102))

Tc103
BNPM beta_negative_decay(Tc103)
AEM (BNPM beta_negative_decay(Tc103))

Tc104
BNPM beta_negative_decay(Tc104)
AEM (BNPM beta_negative_decay(Tc104))

Tc105
BNPM beta_negative_decay(Tc105)
AEM (BNPM beta_negative_decay(Tc105))

Tc106
BNPM beta_negative_decay(Tc106)
AEM (BNPM beta_negative_decay(Tc106))

Tc107
BNPM beta_negative_decay(Tc107)
AEM (BNPM beta_negative_decay(Tc107))

Tc108
BNPM beta_negative_decay(Tc108)
AEM (BNPM beta_negative_decay(Tc108))

Tc109
BNPM beta_negative_decay(Tc109)
AEM (BNPM beta_negative_decay(Tc109))

Tc110
BNPM beta_negative_decay(Tc110)
AEM (BNPM beta_negative_decay(Tc110))

Tc111
BNPM beta_negative_decay(Tc111)
AEM (BNPM beta_negative_decay(Tc111))
```

```
Mo100
BNPM radioactive_capture(Mo100,neutron)
AEM (BNPM radioactive_capture(Mo100,neutron))

Mo101
BNPM beta_negative_decay(Mo101)
AEM (BNPM beta_negative_decay(Mo101))

Mo102
BNPM beta_negative_decay(Mo102)
AEM (BNPM beta_negative_decay(Mo102))

Mo103
BNPM beta_negative_decay(Mo103)
AEM (BNPM beta_negative_decay(Mo103))

Mo104
BNPM beta_negative_decay(Mo104)
AEM (BNPM beta_negative_decay(Mo104))

Mo105
BNPM beta_negative_decay(Mo105)
AEM (BNPM beta_negative_decay(Mo105))

Mo106
BNPM beta_negative_decay(Mo106)
AEM (BNPM beta_negative_decay(Mo106))

Mo107
BNPM beta_negative_decay(Mo107)
AEM (BNPM beta_negative_decay(Mo107))

Mo108

Nb97
BNPM beta_negative_decay(Nb97)
AEM (BNPM beta_negative_decay(Nb97))

Nb98
BNPM beta_negative_decay(Nb98)
AEM (BNPM beta_negative_decay(Nb98))
```

```
Nb99
BNPM beta_negative_decay(Nb99)
AEM (BNPM beta_negative_decay(Nb99))

Nb100
BNPM beta_negative_decay(Nb100)
AEM (BNPM beta_negative_decay(Nb100))

Nb101
BNPM beta_negative_decay(Nb101)
AEM (BNPM beta_negative_decay(Nb101))

Nb102
BNPM beta_negative_decay(Nb102)
AEM (BNPM beta_negative_decay(Nb102))

Nb103
BNPM beta_negative_decay(Nb103)
AEM (BNPM beta_negative_decay(Nb103))

Nb104
BNPM beta_negative_decay(Nb104)
AEM (BNPM beta_negative_decay(Nb104))

Nb105
BNPM beta_negative_decay(Nb105)
AEM (BNPM beta_negative_decay(Nb105))

Nb106
BNPM beta_negative_decay(Nb106)
AEM (BNPM beta_negative_decay(Nb106))

Zr94

Zr95
BNPM beta_negative_decay(Zr95)
AEM (BNPM beta_negative_decay(Zr95))

Zr96

Zr97
BNPM beta_negative_decay(Zr97)
```

```
AEM (BNPM beta_negative_decay(Zr97))

Zr98
BNPM beta_negative_decay(Zr98)
AEM (BNPM beta_negative_decay(Zr98))

Zr99
BNPM beta_negative_decay(Zr99)
AEM (BNPM beta_negative_decay(Zr99))

Zr100
BNPM beta_negative_decay(Zr100)
AEM (BNPM beta_negative_decay(Zr100))

Zr101
BNPM beta_negative_decay(Zr101)
AEM (BNPM beta_negative_decay(Zr101))

Zr102
BNPM beta_negative_decay(Zr102)
AEM (BNPM beta_negative_decay(Zr102))

Zr103
BNPM beta_negative_decay(Zr103)
AEM (BNPM beta_negative_decay(Zr103))

Zr104
BNPM beta_negative_decay(Zr104)
AEM (BNPM beta_negative_decay(Zr104))

Y92
BNPM beta_negative_decay(Y92)
AEM (BNPM beta_negative_decay(Y92))

Y93
BNPM beta_negative_decay(Y93)
AEM (BNPM beta_negative_decay(Y93))

Y94
BNPM beta_negative_decay(Y94)
AEM (BNPM beta_negative_decay(Y94))
```

```
Y95
BNPM beta_negative_decay(Y95)
AEM (BNPM beta_negative_decay(Y95))

Y96
BNPM beta_negative_decay(Y96)
AEM (BNPM beta_negative_decay(Y96))

Y97
BNPM beta_negative_decay(Y97)
AEM (BNPM beta_negative_decay(Y97))

Y98
BNPM beta_negative_decay(Y98)
AEM (BNPM beta_negative_decay(Y98))

Y99
BNPM beta_negative_decay(Y99)
AEM (BNPM beta_negative_decay(Y99))

Y100
BNPM beta_negative_decay(Y100)
AEM (BNPM beta_negative_decay(Y100))

Y101
BNPM beta_negative_decay(Y101)
AEM (BNPM beta_negative_decay(Y101))

Y102
BNPM beta_negative_decay(Y102)
AEM (BNPM beta_negative_decay(Y102))

Sr89
BNPM beta_negative_decay(Sr89)
AEM (BNPM beta_negative_decay(Sr89))

Sr90
BNPM beta_negative_decay(Sr90)
AEM (BNPM beta_negative_decay(Sr90))

Sr91
BNPM beta_negative_decay(Sr91)
```

```
AEM (BNPM beta_negative_decay(Sr91))

Sr92
BNPM beta_negative_decay(Sr92)
AEM (BNPM beta_negative_decay(Sr92))

Sr93
BNPM beta_negative_decay(Sr93)
AEM (BNPM beta_negative_decay(Sr93))

Sr94
BNPM beta_negative_decay(Sr94)
AEM (BNPM beta_negative_decay(Sr94))

Sr95
BNPM beta_negative_decay(Sr95)
AEM (BNPM beta_negative_decay(Sr95))

Sr96
BNPM beta_negative_decay(Sr96)
AEM (BNPM beta_negative_decay(Sr96))

Sr97
BNPM beta_negative_decay(Sr97)
AEM (BNPM beta_negative_decay(Sr97))

Sr98
BNPM beta_negative_decay(Sr98)
AEM (BNPM beta_negative_decay(Sr98))

Sr99
BNPM beta_negative_decay(Sr99)
AEM (BNPM beta_negative_decay(Sr99))

Sr100
BNPM beta_negative_decay(Sr100)
AEM (BNPM beta_negative_decay(Sr100))

Sr101
BNPM beta_negative_decay(Sr101)
AEM (BNPM beta_negative_decay(Sr101))
```

```
Sr102
BNPM beta_negative_decay(Sr102)
AEM (BNPM beta_negative_decay(Sr102))

Rb87

Rb88
BNPM beta_negative_decay(Rb88)
AEM (BNPM beta_negative_decay(Rb88))

Rb89
BNPM beta_negative_decay(Rb89)
AEM (BNPM beta_negative_decay(Rb89))

Rb90
BNPM beta_negative_decay(Rb90)
AEM (BNPM beta_negative_decay(Rb90))

Rb91
BNPM beta_negative_decay(Rb91)
AEM (BNPM beta_negative_decay(Rb91))

Rb92
BNPM beta_negative_decay(Rb92)
AEM (BNPM beta_negative_decay(Rb92))

Rb93
BNPM beta_negative_decay(Rb93)
AEM (BNPM beta_negative_decay(Rb93))

Rb94
BNPM beta_negative_decay(Rb94)
AEM (BNPM beta_negative_decay(Rb94))

Rb95
BNPM beta_negative_decay(Rb95)
AEM (BNPM beta_negative_decay(Rb95))

Rb96
BNPM beta_negative_decay(Rb96)
AEM (BNPM beta_negative_decay(Rb96))
```

```
Rb97
BNPM beta_negative_decay(Rb97)
AEM (BNPM beta_negative_decay(Rb97))

Rb98
BNPM beta_negative_decay(Rb98)
AEM (BNPM beta_negative_decay(Rb98))

Rb99
BNPM beta_negative_decay(Rb99)
AEM (BNPM beta_negative_decay(Rb99))

Rb100
BNPM beta_negative_decay(Rb100)
AEM (BNPM beta_negative_decay(Rb100))

Rb101

Rb102
BNPM beta_negative_decay(Rb102)
AEM (BNPM beta_negative_decay(Rb102))

Kr84

Kr85
BNPM radioactive_capture(Kr85,neutron)
AEM (BNPM radioactive_capture(Kr85,neutron))
BNPM beta_negative_decay(Kr85)
AEM (BNPM beta_negative_decay(Kr85))

Kr86

Kr87
BNPM beta_negative_decay(Kr87)
AEM (BNPM beta_negative_decay(Kr87))

Kr88
BNPM beta_negative_decay(Kr88)
AEM (BNPM beta_negative_decay(Kr88))

Kr89
BNPM beta_negative_decay(Kr89)
```

```
AEM (BNPM beta_negative_decay(Kr89))

Kr90
BNPM beta_negative_decay(Kr90)
AEM (BNPM beta_negative_decay(Kr90))

Kr91
BNPM beta_negative_decay(Kr91)
AEM (BNPM beta_negative_decay(Kr91))

Kr92
BNPM beta_negative_decay(Kr92)
AEM (BNPM beta_negative_decay(Kr92))

Kr93
BNPM beta_negative_decay(Kr93)
AEM (BNPM beta_negative_decay(Kr93))

Kr94
BNPM beta_negative_decay(Kr94)
AEM (BNPM beta_negative_decay(Kr94))

Kr95
BNPM beta_negative_decay(Kr95)
AEM (BNPM beta_negative_decay(Kr95))

Kr96

Kr97
BNPM beta_negative_decay(Kr97)
AEM (BNPM beta_negative_decay(Kr97))

Br82
BNPM beta_negative_decay(Br82)
AEM (BNPM beta_negative_decay(Br82))

Br83
BNPM beta_negative_decay(Br83)
AEM (BNPM beta_negative_decay(Br83))

Br84
BNPM beta_negative_decay(Br84)
```

```
AEM (BNPM beta_negative_decay(Br84))

Br85
BNPM beta_negative_decay(Br85)
AEM (BNPM beta_negative_decay(Br85))

Br86
BNPM beta_negative_decay(Br86)
AEM (BNPM beta_negative_decay(Br86))

Br87
BNPM beta_negative_decay(Br87)
AEM (BNPM beta_negative_decay(Br87))

Br88
BNPM beta_negative_decay(Br88)
AEM (BNPM beta_negative_decay(Br88))

Br89
BNPM beta_negative_decay(Br89)
AEM (BNPM beta_negative_decay(Br89))

Br90
BNPM beta_negative_decay(Br90)
AEM (BNPM beta_negative_decay(Br90))

Br91
BNPM beta_negative_decay(Br91)
AEM (BNPM beta_negative_decay(Br91))

Br92
BNPM beta_negative_decay(Br92)
AEM (BNPM beta_negative_decay(Br92))

Br93

Br94

Se79

Se80
```

```
Se81
BNPM beta_negative_decay(Se81)
AEM (BNPM beta_negative_decay(Se81))

Se82

Se83
BNPM beta_negative_decay(Se83)
AEM (BNPM beta_negative_decay(Se83))

Se84
BNPM beta_negative_decay(Se84)
AEM (BNPM beta_negative_decay(Se84))

Se85
BNPM beta_negative_decay(Se85)
AEM (BNPM beta_negative_decay(Se85))

Se86
BNPM beta_negative_decay(Se86)
AEM (BNPM beta_negative_decay(Se86))

Se87
BNPM beta_negative_decay(Se87)
AEM (BNPM beta_negative_decay(Se87))

Se88
BNPM beta_negative_decay(Se88)
AEM (BNPM beta_negative_decay(Se88))

Se89
BNPM beta_negative_decay(Se89)
AEM (BNPM beta_negative_decay(Se89))

Se90

Se91
BNPM beta_negative_decay(Se91)
AEM (BNPM beta_negative_decay(Se91))

As77
BNPM beta_negative_decay(As77)
```

```
AEM (BNPM beta_negative_decay(As77))

As78
BNPM beta_negative_decay(As78)
AEM (BNPM beta_negative_decay(As78))

As79
BNPM beta_negative_decay(As79)
AEM (BNPM beta_negative_decay(As79))

As80
BNPM beta_negative_decay(As80)
AEM (BNPM beta_negative_decay(As80))

As81
BNPM beta_negative_decay(As81)
AEM (BNPM beta_negative_decay(As81))

As82
BNPM beta_negative_decay(As82)
AEM (BNPM beta_negative_decay(As82))

As83
BNPM beta_negative_decay(As83)
AEM (BNPM beta_negative_decay(As83))

As84
BNPM beta_negative_decay(As84)
AEM (BNPM beta_negative_decay(As84))

As85
BNPM beta_negative_decay(As85)
AEM (BNPM beta_negative_decay(As85))

As86
BNPM beta_negative_decay(As86)
AEM (BNPM beta_negative_decay(As86))

As87
BNPM beta_negative_decay(As87)
AEM (BNPM beta_negative_decay(As87))
```

```
Ge74

Ge75
BNPM beta_negative_decay(Ge75)
AEM (BNPM beta_negative_decay(Ge75))

Ge76

Ge77
BNPM beta_negative_decay(Ge77)
AEM (BNPM beta_negative_decay(Ge77))

Ge78
BNPM beta_negative_decay(Ge78)
AEM (BNPM beta_negative_decay(Ge78))

Ge79
BNPM beta_negative_decay(Ge79)
AEM (BNPM beta_negative_decay(Ge79))

Ge80
BNPM beta_negative_decay(Ge80)
AEM (BNPM beta_negative_decay(Ge80))

Ge81
BNPM beta_negative_decay(Ge81)
AEM (BNPM beta_negative_decay(Ge81))

Ge82
BNPM beta_negative_decay(Ge82)
AEM (BNPM beta_negative_decay(Ge82))

Ge83
BNPM beta_negative_decay(Ge83)
AEM (BNPM beta_negative_decay(Ge83))

Ge84
BNPM beta_negative_decay(Ge84)
AEM (BNPM beta_negative_decay(Ge84))

Ga72
BNPM beta_negative_decay(Ga72)
```

```
AEM (BNPM beta_negative_decay(Ga72))

Ga73
BNPM beta_negative_decay(Ga73)
AEM (BNPM beta_negative_decay(Ga73))

Ga74
BNPM beta_negative_decay(Ga74)
AEM (BNPM beta_negative_decay(Ga74))

Ga75
BNPM beta_negative_decay(Ga75)
AEM (BNPM beta_negative_decay(Ga75))

Ga76
BNPM beta_negative_decay(Ga76)
AEM (BNPM beta_negative_decay(Ga76))

Ga77
BNPM beta_negative_decay(Ga77)
AEM (BNPM beta_negative_decay(Ga77))

Ga78
BNPM beta_negative_decay(Ga78)
AEM (BNPM beta_negative_decay(Ga78))

Ga79
BNPM beta_negative_decay(Ga79)
AEM (BNPM beta_negative_decay(Ga79))

Ga80
BNPM beta_negative_decay(Ga80)
AEM (BNPM beta_negative_decay(Ga80))

Ga81
BNPM beta_negative_decay(Ga81)
AEM (BNPM beta_negative_decay(Ga81))

Ga82
BNPM beta_negative_decay(Ga82)
AEM (BNPM beta_negative_decay(Ga82))
```

```
Ga83
BNPM beta_negative_decay(Ga83)
AEM (BNPM beta_negative_decay(Ga83))

Zn72
BNPM beta_negative_decay(Zn72)
AEM (BNPM beta_negative_decay(Zn72))

Zn73
BNPM beta_negative_decay(Zn73)
AEM (BNPM beta_negative_decay(Zn73))

Zn74
BNPM beta_negative_decay(Zn74)
AEM (BNPM beta_negative_decay(Zn74))

Zn75
BNPM beta_negative_decay(Zn75)
AEM (BNPM beta_negative_decay(Zn75))

Zn76
BNPM beta_negative_decay(Zn76)
AEM (BNPM beta_negative_decay(Zn76))

Zn77
BNPM beta_negative_decay(Zn77)
AEM (BNPM beta_negative_decay(Zn77))

Zn78
BNPM beta_negative_decay(Zn78)
AEM (BNPM beta_negative_decay(Zn78))

Zn79
BNPM beta_negative_decay(Zn79)
AEM (BNPM beta_negative_decay(Zn79))

Zn80
BNPM beta_negative_decay(Zn80)
AEM (BNPM beta_negative_decay(Zn80))

Cu72
BNPM beta_negative_decay(Cu72)
```

```
AEM (BNPM beta_negative_decay(Cu72))

Cu73
BNPM beta_negative_decay(Cu73)
AEM (BNPM beta_negative_decay(Cu73))

Cu74

Cu75
BNPM beta_negative_decay(Cu75)
AEM (BNPM beta_negative_decay(Cu75))

Cu76
BNPM beta_negative_decay(Cu76)
AEM (BNPM beta_negative_decay(Cu76))
```

Appendix V

State and Probabilistic Space Equations at the End of the Second Cycle

This appendix shows the probabilistic space and the state space equations at the end of the second cycle (t_3) of running the simulation for Experiment No.2.

```
Sigma_a_fuel = U235.amount*U235.sigma_g+U235.amount*U235.sigma_f+
U236.amount*U236.sigma_g+U238.amount*U238.sigma_g+Tb159.amount*
Tb159.sigma_g+Gd155.amount*Gd155.sigma_g+Gd156.amount*Gd156.sigma_g+
Gd157.amount*Gd157.sigma_g+Eu153.amount*Eu153.sigma_g+
Eu154.amount*Eu154.sigma_g+Eu155.amount*Eu155.sigma_g+
Sm150.amount*Sm150.sigma_g+Sm151.amount*Sm151.sigma_g+
Sm152.amount*Sm152.sigma_g+Sm153.amount*Sm153.sigma_g+
Sm154.amount*Sm154.sigma_g+Pm148.amount*Pm148.sigma_g+
Nd145.amount*Nd145.sigma_g+Nd146.amount*Nd146.sigma_g+
Nd148.amount*Nd148.sigma_g+Nd150.amount*Nd150.sigma_g+
Pr143.amount*Pr143.sigma_g+La138.amount*La138.sigma_g+
Cs133.amount*Cs133.sigma_g+Xe130.amount*Xe130.sigma_g+
Xe131.amount*Xe131.sigma_g+Xe135.amount*Xe135.sigma_g+
I127.amount*I127.sigma_g+I129.amount*I129.sigma_g+
Sb123.amount*Sb123.sigma_g+Mo100.amount*Mo100.sigma_g+
Kr85.amount*Kr85.sigma_g;

FgU235 = U235.amount*U235.sigma_g/Sigma_a_fuel;

nrcU235 = neutron.amount*k/eta*FgU235;

PrcU235 = nrcU235*U235.A/NA;

CrcU235 = nrcU235*U235.A/NA;

U235.amount += -1.00*CrcU235-CfU235;

U236.amount += 1.00*PrcU235-1.00*CrcU236;

Gamma.energy += 1.98*nrcU235+sum((i+ei/2)*distr_E_GammaU235[i],0,8)+
1.98*nrcU236+1.98*nrcU238+1.98*nrcTb159+1.98*nrcGd155+1.98*nrcGd156+
1.98*nrcGd157+1.98*nrcEu153+1.98*nrcEu154+1.98*nrcEu155+1.98*nrcSm150+
1.98*nrcSm151+1.98*nrcSm152+1.98*nrcSm153+1.98*nrcSm154+1.98*nrcPm148+
1.98*nrcNd145+1.98*nrcNd146+1.98*nrcNd148+1.98*nrcNd150+1.98*nrcPr143+
1.98*nrcLa138+1.98*nrcCs133+1.98*nrcXe130+1.98*nrcXe131+1.98*nrcXe135+
1.98*nrcI127+1.98*nrcI129+1.98*nrcSb123+1.98*nrcMo100+1.98*nrcKr85;

Sigma_f_fuel = U235.amount*U235.sigma_f;

fU235 = U235.amount*(U235.sigma_f+U235.sigma_g)/
(Sigma_a_fuel+Sigma_a_cl+Sigma_a_md);

eta = nuU235*U235.amount*U235.sigma_f/Sigma_a_fuel;

k = fU235*eta*RC.epsilon*RC.p*RC.PNLf*RC.PNLth;

FfU235 = U235.amount*U235.sigma_f/Sigma_a_fuel;

RcU235 = k/eta*neutron.amount*FfU235;

CfU235 = RcU235/NA*U235.A;

distr_neutronU235[i] = RcU235*pdf_neutronU235[i],0,8;

pnU235 = sum(i*distr_neutronU235[i],0,8);

distr_A1U235[i] = RcU235*pdf_A1U235[i],0,44;

distr_A2U235[i] = RcU235*pdf_A2U235[i],0,44;

distr_E_neutronU235[i] = pnU235*pdf_E_neutronU235[i],0,8;
```

```
distr_E_gammaU235[i] = RcU235*pdf_E_gammaU235[i],0,8;
distr_E_F1U235[i] = RcU235*pdf_E_F1U235[i],0,44;
distr_E_F2U235[i] = RcU235*pdf_E_F2U235[i],0,44;
Fissile_fissioned += RcU235;
Fissile_consumed += CfU235;
Neutron.amount += pnU235;
Neutron.energy += sum((i+ei/2)*distr_E_neutronU235[i],0,8);
Energy_F1.amount += sum((i+ei/2)*distr_E_F1U235[i],0,90);
Energy_F2.amount += sum((i+ei/2)*distr_E_F2U235[i],0,90);
FgU236 = U236.amount*U236.sigma_g/Sigma_a_fuel;
nrcU236 = neutron.amount*k/eta*FgU236;
PrcU236 = nrcU236*U236.A/NA;
CrcU236 = nrcU236*U236.A/NA;
U237.amount += 1.00*PrcU236;
FgU238 = U238.amount*U238.sigma_g/Sigma_a_fuel;
nrcU238 = neutron.amount*k/eta*FgU238;
PrcU238 = nrcU238*U238.A/NA;
CrcU238 = nrcU238*U238.A/NA;
U238.amount += -1.00*CrcU238;
U239.amount += 1.00*PrcU238;
PbndU239 = ld*U239.halflife*U239.amount*NA/U239.A*cycle_time;
CbndU239 = ld*U239.halflife*U239.amount*NA/U239.A*cycle_time;
Np239.amount += 1.00*PbndU239/NA*U239.A;
Beta_Neg.amount += 1.00*PbndU239+1.00*PbndTb158+1.00*PbndGd159+
1.00*PbndEu155+1.00*PbndEu156+1.00*PbndEu157+1.00*PbndEu158+
1.00*PbndEu159+1.00*PbndSm151+1.00*PbndSm153+1.00*PbndSm155+
1.00*PbndSm156+1.00*PbndSm157+1.00*PbndSm158+1.00*PbndSm159+
1.00*PbndPm148+1.00*PbndPm149+1.00*PbndPm150+1.00*PbndPm151+
1.00*PbndPm152+1.00*PbndPm153+1.00*PbndPm154+1.00*PbndPm155+
1.00*PbndPm156+1.00*PbndPm157+1.00*PbndPm158+1.00*PbndNd147+
1.00*PbndNd149+1.00*PbndNd151+1.00*PbndNd152+1.00*PbndNd153+
1.00*PbndNd154+1.00*PbndNd155+1.00*PbndNd156+1.00*PbndPr143+
1.00*PbndPr144+1.00*PbndPr145+1.00*PbndPr146+1.00*PbndPr147+
1.00*PbndPr148+1.00*PbndPr149+1.00*PbndPr150+1.00*PbndPr151+
1.00*PbndPr152+1.00*PbndPr153+1.00*PbndPr154+1.00*PbndCe141+
1.00*PbndCe143+1.00*PbndCe144+1.00*PbndCe145+1.00*PbndCe146+
1.00*PbndCe147+1.00*PbndCe148+1.00*PbndCe149+1.00*PbndCe150+
1.00*PbndCe151+1.00*PbndCe152+1.00*PbndLa140+1.00*PbndLa141+
1.00*PbndLa142+1.00*PbndLa143+1.00*PbndLa144+1.00*PbndLa145+
```

```
1.00*PbndLa146+1.00*PbndLa147+1.00*PbndLa148+1.00*PbndLa149+
1.00*PbndBa139+1.00*PbndBa140+1.00*PbndBa141+1.00*PbndBa142+
1.00*PbndBa143+1.00*PbndBa144+1.00*PbndBa145+1.00*PbndBa146+
1.00*PbndBa147+1.00*PbndBa148+1.00*PbndBa149+1.00*PbndCs134+
1.00*PbndCs136+1.00*PbndCs137+1.00*PbndCs138+1.00*PbndCs139+
1.00*PbndCs140+1.00*PbndCs141+1.00*PbndCs142+1.00*PbndCs143+
1.00*PbndCs144+1.00*PbndCs145+1.00*PbndCs146+1.00*PbndCs147+
1.00*PbndCs148+1.00*PbndXe133+1.00*PbndXe135+1.00*PbndXe137+
1.00*PbndXe138+1.00*PbndXe139+1.00*PbndXe140+1.00*PbndXe141+
1.00*PbndXe142+1.00*PbndXe143+1.0*PbndXe144+1.00*PbndXe145+
1.00*PbndI128+1.00*PbndI130+1.00*PbndI131+1.00*PbndI132+
1.00*PbndI133+1.00*PbndI134+1.00*PbndI135+1.00*PbndI136+
1.00*PbndI137+1.00*PbndI138+1.00*PbndI139+1.00*PbndI140+
1.00*PbndI141+1.00*PbndI142+1.00*PbndTe127+1.00*PbndTe129+
1.00*PbndTe131+1.00*PbndTe132+1.00*PbndTe133+1.00*PbndTe134+
1.00*PbndTe135+1.00*PbndTe136+1.00*PbndTe137+1.00*PbndTe138+
1.00*PbndSb122+1.00*PbndSb124+1.00*PbndSb125+1.00*PbndSb126+
1.00*PbndSb127+1.00*PbndSb128+1.00*PbndSb129+1.00*PbndSb130+
1.00*PbndSb131+1.00*PbndSb132+1.00*PbndSb133+1.00*PbndSb134+
1.00*PbndSb135+1.00*PbndSb136+1.00*PbndSn121+1.00*PbndSn123+
1.00*PbndSn125+1.00*PbndSn127+1.00*PbndSn128+1.00*PbndSn129+
1.00*PbndSn130+1.00*PbndSn131+1.00*PbndSn132+1.00*PbndSn133+
1.00*PbndSn134+1.00*PbndIn117+1.00*PbndIn118+1.00*PbndIn119+
1.00*PbndIn120+1.00*PbndIn121+1.00*PbndIn122+1.00*PbndIn123+
1.00*PbndIn124+1.00*PbndIn125+1.00*PbndIn126+1.00*PbndIn127+
1.00*PbndIn128+1.00*PbndIn129+1.00*PbndIn130+1.00*PbndIn131+
1.00*PbndIn132+1.00*PbndIn133+1.00*PbndCd115+1.00*PbndCd117+
1.00*PbndCd118+1.00*PbndCd119+1.00*PbndCd120+1.00*PbndCd121+
1.00*PbndCd122+1.00*PbndCd123+1.00*PbndCd124+1.00*PbndCd125+
1.00*PbndCd126+1.00*PbndCd127+1.00*PbndCd128+1.00*PbndCd129+
1.00*PbndCd130+1.00*PbndAg112+1.00*PbndAg113+1.00*PbndAg114+
1.00*PbndAg115+1.00*PbndAg116+1.00*PbndAg117+1.00*PbndAg118+
1.00*PbndAg119+1.00*PbndAg120+1.00*PbndAg121+1.00*PbndAg122+
1.00*PbndAg123+1.00*PbndAg124+1.00*PbndPd111+1.00*PbndPd112+
1.00*PbndPd113+1.00*PbndPd114+1.00*PbndPd115+1.00*PbndPd116+
1.00*PbndPd117+1.00*PbndPd118+1.00*PbndRh107+1.00*PbndRh108+
1.00*PbndRh109+1.00*PbndRh110+1.00*PbndRh111+1.00*PbndRh112+
1.00*PbndRh113+1.00*PbndRh114+1.00*PbndRh115+1.00*PbndRh116+
1.00*PbndRu105+1.00*PbndRu106+1.00*PbndRu107+1.00*PbndRu108+
1.00*PbndRu109+1.00*PbndRu110+1.00*PbndRu111+1.00*PbndRu112+
1.00*PbndRu113+1.00*PbndTc102+1.00*PbndTc103+1.00*PbndTc104+
1.00*PbndTc105+1.00*PbndTc106+1.00*PbndTc107+1.00*PbndTc108+
1.00*PbndTc109+1.00*PbndTc110+1.00*PbndTc111+1.00*PbndMo101+
1.00*PbndMo102+1.00*PbndMo103+1.00*PbndMo104+1.00*PbndMo105+
1.00*PbndMo106+1.00*PbndMo107+1.00*PbndNb97+1.00*PbndNb98+
1.00*PbndNb99+1.00*PbndNb100+1.00*PbndNb101+1.00*PbndNb102+
1.00*PbndNb103+1.00*PbndNb104+1.00*PbndNb105+1.00*PbndNb106+
1.00*PbndZr95+1.00*PbndZr97+1.00*PbndZr98+1.00*PbndZr99+
1.00*PbndZr100+1.00*PbndZr101+1.00*PbndZr102+1.00*PbndZr103+
1.00*PbndZr104+1.00*PbndY92+1.00*PbndY93+1.00*PbndY94+
1.00*PbndY95+1.00*PbndY96+1.00*PbndY97+1.00*PbndY98+
1.00*PbndY99+1.00*PbndY100+1.00*PbndY101+1.00*PbndY102+
1.00*PbndSr89+1.00*PbndSr90+1.00*PbndSr91+1.00*PbndSr92+
1.00*PbndSr93+1.00*PbndSr94+1.00*PbndSr95+1.00*PbndSr96+
1.00*PbndSr97+1.00*PbndSr98+1.00*PbndSr99+1.00*PbndSr100+
1.00*PbndSr101+1.00*PbndSr102+1.00*PbndRb88+1.00*PbndRb89+
1.00*PbndRb90+1.00*PbndRb91+1.00*PbndRb92+1.00*PbndRb93+
1.00*PbndRb94+1.00*PbndRb95+1.00*PbndRb96+1.00*PbndRb97+
1.00*PbndRb98+1.00*PbndRb99+1.00*PbndRb100+1.00*PbndRb102+
1.00*PbndKr85+1.00*PbndKr87+1.00*PbndKr88+1.00*PbndKr89+
1.00*PbndKr90+1.00*PbndKr91+1.00*PbndKr92+1.00*PbndKr93+
1.00*PbndKr94+1.00*PbndKr95+1.00*PbndKr97+1.00*PbndBr82+
1.00*PbndBr83+1.00*PbndBr84+1.00*PbndBr85+1.00*PbndBr86+
```

```
1.00*PbndBr87+1.00*PbndBr88+1.00*PbndBr89+1.00*PbndBr90+
1.00*PbndBr91+1.00*PbndBr92+1.00*PbndSe81+1.00*PbndSe83+
1.00*PbndSe84+1.00*PbndSe85+1.00*PbndSe86+1.00*PbndSe87+
1.00*PbndSe88+1.00*PbndSe89+1.00*PbndSe91+1.00*PbndAs77+
1.00*PbndAs78+1.00*PbndAs79+1.00*PbndAs80+1.00*PbndAs81+
1.00*PbndAs82+1.00*PbndAs83+1.00*PbndAs84+1.00*PbndAs85+
1.00*PbndAs86+1.00*PbndAs87+1.00*PbndGe75+1.00*PbndGe77+
1.00*PbndGe78+1.00*PbndGe79+1.00*PbndGe80+1.00*PbndGe81+
1.00*PbndGe82+1.00*PbndGe83+1.00*PbndGe84+1.00*PbndGa72+
1.00*PbndGa73+1.00*PbndGa74+1.00*PbndGa75+1.00*PbndGa76+
1.00*PbndGa77+1.00*PbndGa78+1.00*PbndGa79+1.00*PbndGa80+
1.00*PbndGa81+1.00*PbndGa82+1.00*PbndGa83+1.00*PbndZn72+
1.00*PbndZn73+1.00*PbndZn74+1.00*PbndZn75+1.00*PbndZn76+
1.00*PbndZn77+1.00*PbndZn78+1.00*PbndZn79+1.00*PbndZn80+
1.00*PbndCu72+1.00*PbndCu73+1.00*PbndCu75+1.00*PbndCu76;

Beta_Neg.energy += 0.40*PbndU239+0.40*PbndTb158+0.40*PbndGd159+
0.40*PbndEu155+0.40*PbndEu156+0.40*PbndEu157+0.40*PbndEu158+
0.40*PbndEu159+0.40*PbndSm151+0.40*PbndSm153+0.40*PbndSm155+
0.40*PbndSm156+0.40*PbndSm157+0.40*PbndSm158+0.40*PbndSm159+
0.40*PbndPm148+0.40*PbndPm149+0.40*PbndPm150+0.40*PbndPm151+
0.40*PbndPm152+0.40*PbndPm153+0.40*PbndPm154+0.40*PbndPm155+
0.40*PbndPm156+0.40*PbndPm157+0.40*PbndPm158+0.40*PbndNd147+
0.40*PbndNd149+0.40*PbndNd151+0.40*PbndNd152+0.40*PbndNd153+
0.40*PbndNd154+0.40*PbndNd155+0.40*PbndNd156+0.40*PbndPr143+
0.40*PbndPr144+0.40*PbndPr145+0.40*PbndPr146+0.40*PbndPr147+
0.40*PbndPr148+0.40*PbndPr149+0.40*PbndPr150+0.40*PbndPr151+
0.40*PbndPr152+0.40*PbndPr153+0.40*PbndPr154+0.40*PbndCe141+
0.40*PbndCe143+0.40*PbndCe144+0.40*PbndCe145+0.40*PbndCe146+
0.40*PbndCe147+0.40*PbndCe148+0.40*PbndCe149+0.40*PbndCe150+
0.40*PbndCe151+0.40*PbndCe152+0.40*PbndLa140+0.40*PbndLa141+
0.40*PbndLa142+0.40*PbndLa143+0.40*PbndLa144+0.40*PbndLa145+
0.40*PbndLa146+0.40*PbndLa147+0.40*PbndLa148+0.40*PbndLa149+
0.40*PbndBa139+0.40*PbndBa140+0.40*PbndBa141+0.40*PbndBa142+
0.40*PbndBa143+0.40*PbndBa144+0.40*PbndBa145+0.40*PbndBa146+
0.40*PbndBa147+0.40*PbndBa148+0.40*PbndBa149+0.40*PbndCs134+
0.40*PbndCs136+0.40*PbndCs137+0.40*PbndCs138+0.40*PbndCs139+
0.40*PbndCs140+0.40*PbndCs141+0.40*PbndCs142+0.40*PbndCs143+
0.40*PbndCs144+0.40*PbndCs145+0.40*PbndCs146+0.40*PbndCs147+
0.40*PbndCs148+0.40*PbndXe133+0.40*PbndXe135+0.40*PbndXe137+
0.40*PbndXe138+0.40*PbndXe139+0.40*PbndXe140+0.40*PbndXe141+
0.40*PbndXe142+0.40*PbndXe143+0.40*PbndXe144+0.40*PbndXe145+
0.40*PbndI128+0.40*PbndI130+0.40*PbndI131+0.40*PbndI132+
0.40*PbndI133+0.40*PbndI134+0.40*PbndI135+0.40*PbndI136+
0.40*PbndI137+0.40*PbndI138+0.40*PbndI139+0.40*PbndI140+
0.40*PbndI141+0.40*PbndI142+0.40*PbndTe127+0.40*PbndTe129+
0.40*PbndTe131+0.40*PbndTe132+0.40*PbndTe133+0.40*PbndTe134+
0.40*PbndTe135+0.40*PbndTe136+0.40*PbndTe137+0.40*PbndTe138+
0.40*PbndSb122+0.40*PbndSb124+0.40*PbndSb125+0.40*PbndSb126+
0.40*PbndSb127+0.40*PbndSb128+0.40*PbndSb129+0.40*PbndSb130+
0.40*PbndSb131+0.40*PbndSb132+0.40*PbndSb133+0.40*PbndSb134+
0.40*PbndSb135+0.40*PbndSb136+0.40*PbndSn121+0.40*PbndSn123+
0.40*PbndSn125+0.40*PbndSn127+0.40*PbndSn128+0.40*PbndSn129+
0.40*PbndSn130+0.40*PbndSn131+0.40*PbndSn132+0.40*PbndSn133+
0.40*PbndSn134+0.40*PbndIn117+0.40*PbndIn118+0.40*PbndIn119+
0.40*PbndIn120+0.40*PbndIn121+0.40*PbndIn122+0.40*PbndIn123+
0.40*PbndIn130+0.40*PbndIn131+0.40*PbndIn132+0.40*PbndIn133+
0.40*PbndCd115+0.40*PbndCd117+0.40*PbndCd118+0.40*PbndCd119+
0.40*PbndCd120+0.40*PbndCd121+0.40*PbndCd122+0.40*PbndCd123+
0.40*PbndCd124+0.40*PbndCd125+0.40*PbndCd126+0.40*PbndCd127+
0.40*PbndCd128+0.40*PbndCd129+0.40*PbndCd130+0.40*PbndAg112+
0.40*PbndAg113+0.40*PbndAg114+0.40*PbndAg115+0.40*PbndAg116+
0.40*PbndAg117+0.40*PbndAg118+0.40*PbndAg119+0.40*PbndAg120+
```

```
0.40*PbndAg121+0.40*PbndAg122+0.40*PbndAg123+0.40*PbndAg124+
0.40*PbndPd111+0.40*PbndPd112+0.40*PbndPd113+0.40*PbndPd114+
0.40*PbndPd115+0.40*PbndPd116+0.40*PbndPd117+0.40*PbndPd118+
0.40*PbndRh107+0.40*PbndRh108+0.40*PbndRh109+0.40*PbndRh110+
0.40*PbndRh111+0.40*PbndRh112+0.40*PbndRh113+0.40*PbndRh114+
0.40*PbndRh115+0.40*PbndRh116+0.40*PbndRu105+0.40*PbndRu106+
0.40*PbndRu107+0.40*PbndRu108+0.40*PbndRu109+0.40*bndRu110+
0.40*PbndRu111+0.40*PbndRu112+0.40*PbndRu113+0.40*PbndTc102+
0.40*PbndTc103+0.40*PbndTc104+0.0*PbndTc105+0.40*PbndTc106+
0.40*PbndTc107+0.40*PbndTc108+0.40*PbndTc109+0.40*PbndTc110+
0.40*PbndTc111+0.40*PbndMo101+0.40*PbndMo102+0.40*PbndMo103+
0.40*PbndMo104+0.40*PbndMo105+0.40*PbndMo106+0.40*PbndMo107+
0.40*PbndNb97+0.40*PbndNb98+0.40*PbndNb99+0.40*PbndNb100+
0.40*PbndNb101+0.40*PbndNb102+0.40*PbndNb103+0.40*PbndNb104+
0.40*PbndNb105+0.40*PbndNb106+0.40*PbndZr95+0.40*PbndZr97+
0.40*PbndZr98+0.40*PbndZr99+0.40*PbndZr100+0.40*PbndZr101+
0.40*PbndZr102+0.40*PbndZr103+0.40*PbndZr104+0.40*PbndY92+
0.40*PbndY93+0.40*PbndY94+0.40*PbndY95+0.40*PbndY96+0.40*PbndY97+
0.40*PbndY98+0.40*PbndY99+0.40*PbndY100+0.40*PbndY101+
0.40*PbndY102+0.40*PbndSr89+0.40*PbndSr90+0.40*PbndSr91+
0.40*PbndSr92+0.40*PbndSr93+0.40*PbndSr94+0.40*PbndSr95+
0.40*PbndSr96+0.40*PbndSr97+0.40*PbndSr98+0.40*PbndSr99+
0.40*PbndSr100+0.40*PbndSr101+0.40*PbndSr102+0.40*PbndRb88+
0.40*PbndRb89+0.40*PbndRb90+0.40*PbndRb91+0.40*PbndRb92+
0.40*PbndRb93+0.40*PbndRb94+0.40*PbndRb95+0.40*PbndRb96+
0.40*PbndRb97+0.40*PbndRb98+0.40*PbndRb99+0.40*PbndRb100+
0.40*PbndRb102+0.40*PbndKr85+0.40*PbndKr87+0.40*PbndKr88+
0.40*PbndKr89+0.40*PbndKr90+0.40*PbndKr91+0.40*PbndKr92+
0.40*PbndKr93+0.40*PbndKr94+0.40*PbndKr95+0.40*PbndKr97+
0.40*PbndBr82+0.40*PbndBr83+0.40*PbndBr84+0.40*PbndBr85+
0.40*PbndBr86+0.40*PbndBr87+0.40*PbndBr88+0.40*PbndBr89+
0.40*PbndBr90+0.40*PbndBr91+0.40*PbndBr92+0.40*PbndSe81+
0.40*PbndSe83+0.40*PbndSe84+0.40*PbndSe85+0.40*PbndSe86+
0.40*PbndSe87+0.40*PbndSe88+0.40*PbndSe89+0.40*PbndSe91+
0.40*PbndAs77+0.40*PbndAs78+0.40*PbndAs79+0.40*PbndAs80+
0.40*PbndAs81+0.40*PbndAs82+0.40*PbndAs83+0.40*PbndAs84+
0.40*PbndAs85+0.40*PbndAs86+0.40*PbndAs87+0.40*PbndGe75+
0.40*PbndGe77+0.40*PbndGe78+0.40*PbndGe79+0.40*PbndGe80+
0.40*PbndGe81+0.40*PbndGe82+0.40*PbndGe83+0.40*PbndGe84+
0.40*PbndGa72+0.40*PbndGa73+0.40*PbndGa74+0.40*PbndGa75+
0.40*PbndGa76+0.40*PbndGa77+0.40*PbndGa78+0.40*PbndGa79+
0.40*PbndGa80+0.40*PbndGa81+0.40*PbndGa82+0.40*PbndGa83+
0.40*PbndZn72+0.40*PbndZn73+0.40*PbndZn74+0.40*PbndZn75+
0.40*PbndZn76+0.40*PbndZn77+0.40*PbndZn78+0.40*PbndZn79+
0.40*PbndZn80+0.40*PbndCu72+0.40*PbndCu73+0.40*PbndCu75+
0.40*PbndCu76;

PbndTb158 = ld*Tb158.halflife*Tb158.amount*NA/Tb158.A*cycle_time;

CbndTb158 = ld*Tb158.halflife*Tb158.amount*NA/Tb158.A*cycle_time;

Dy158.amount += 1.00*PbndTb158/NA*Tb158.A;

FgTb159 = Tb159.amount*Tb159.sigma_g/Sigma_a_fuel;

nrcTb159 = neutron.amount*k/eta*FgTb159;

PrcTb159 = nrcTb159*Tb159.A/NA;

CrcTb159 = nrcTb159*Tb159.A/NA;

Tb159.amount += -1.00*CrcTb159+1.00*PbndGd159/NA*Gd159.A;
```

```
Tb160.amount += 1.00*PrcTb159;
FgGd155 = Gd155.amount*Gd155.sigma_g/Sigma_a_fuel;
nrcGd155 = neutron.amount*k/eta*FgGd155;
PrcGd155 = nrcGd155*Gd155.A/NA;
CrcGd155 = nrcGd155*Gd155.A/NA;
Gd155.amount += -1.00*CrcGd155+1.00*PbndEu155/NA*Eu155.A;
Gd156.amount += 1.00*PrcGd155-1.00*CrcGd156+1.00*PbndEu156/NA*Eu156.A;
FgGd156 = Gd156.amount*Gd156.sigma_g/Sigma_a_fuel;
nrcGd156 = neutron.amount*k/eta*FgGd156;
PrcGd156 = nrcGd156*Gd156.A/NA;
CrcGd156 = nrcGd156*Gd156.A/NA;
Gd157.amount += 1.00*PrcGd156-1.00*CrcGd157+1.00*PbndEu157/NA*Eu157.A;
FgGd157 = Gd157.amount*Gd157.sigma_g/Sigma_a_fuel;
nrcGd157 = neutron.amount*k/eta*FgGd157;
PrcGd157 = nrcGd157*Gd157.A/NA;
CrcGd157 = nrcGd157*Gd157.A/NA;
Gd158.amount += 1.00*PrcGd157+1.00*PbndEu158/NA*Eu158.A;
PbndGd159 = ld*Gd159.halflife*Gd159.amount*NA/Gd159.A*cycle_time;
CbndGd159 = ld*Gd159.halflife*Gd159.amount*NA/Gd159.A*cycle_time;
FgEu153 = Eu153.amount*Eu153.sigma_g/Sigma_a_fuel;
nrcEu153 = neutron.amount*k/eta*FgEu153;
PrcEu153 = nrcEu153*Eu153.A/NA;
CrcEu153 = nrcEu153*Eu153.A/NA;
Eu153.amount += -1.00*CrcEu153+1.00*PbndSm153/NA*Sm153.A;
Eu154.amount += 1.00*PrcEu153-1.00*CrcEu154;
FgEu154 = Eu154.amount*Eu154.sigma_g/Sigma_a_fuel;
nrcEu154 = neutron.amount*k/eta*FgEu154;
PrcEu154 = nrcEu154*Eu154.A/NA;
CrcEu154 = nrcEu154*Eu154.A/NA;
Eu155.amount += 1.00*PrcEu154-1.00*CrcEu155+1.00*PbndSm155/NA*Sm155.A;
PbpdEu154 = ld*Eu154.halflife*Eu154.amount*NA/Eu154.A*cycle_time;
CbpdEu154 = ld*Eu154.halflife*Eu154.amount*NA/Eu154.A*cycle_time;
```

```
Sm154.amount += 1.00*PbpdEu154/NA*Eu154.A+1.00*PrcSm153-1.00*
CrcSm154+1.00*PbndPm154/NA*Pm154.A;

Beta_Pos.amount += 1.00*PbpdEu154;

Beta_Pos.energy += 0.40*PbpdEu154;

FgEu155 = Eu155.amount*Eu155.sigma_g/Sigma_a_fuel;

nrcEu155 = neutron.amount*k/eta*FgEu155;

PrcEu155 = nrcEu155*Eu155.A/NA;

CrcEu155 = nrcEu155*Eu155.A/NA;

Eu156.amount += 1.00*PrcEu155+1.00*PbndSm156/NA*Sm156.A;

PbndEu155 = ld*Eu155.halflife*Eu155.amount*NA/Eu155.A*cycle_time;

CbndEu155 = ld*Eu155.halflife*Eu155.amount*NA/Eu155.A*cycle_time;

PbndEu156 = ld*Eu156.halflife*Eu156.amount*NA/Eu156.A*cycle_time;

CbndEu156 = ld*Eu156.halflife*Eu156.amount*NA/Eu156.A*cycle_time;

PbndEu157 = ld*Eu157.halflife*Eu157.amount*NA/Eu157.A*cycle_time;

CbndEu157 = ld*Eu157.halflife*Eu157.amount*NA/Eu157.A*cycle_time;

PbndEu158 = ld*Eu158.halflife*Eu158.amount*NA/Eu158.A*cycle_time;

CbndEu158 = ld*Eu158.halflife*Eu158.amount*NA/Eu158.A*cycle_time;

PbndEu159 = ld*Eu159.halflife*Eu159.amount*NA/Eu159.A*cycle_time;

CbndEu159 = ld*Eu159.halflife*Eu159.amount*NA/Eu159.A*cycle_time;

Gd159.amount += 1.00*PbndEu159/NA*Eu159.A;

FgSm150 = Sm150.amount*Sm150.sigma_g/Sigma_a_fuel;

nrcSm150 = neutron.amount*k/eta*FgSm150;

PrcSm150 = nrcSm150*Sm150.A/NA;

CrcSm150 = nrcSm150*Sm150.A/NA;

Sm150.amount += -1.00*CrcSm150+1.00*PbndPm150/NA*Pm150.A;

Sm151.amount += 1.00*PrcSm150-1.00*CrcSm151+
1.00*PbndPm151/NA*Pm151.A;

FgSm151 = Sm151.amount*Sm151.sigma_g/Sigma_a_fuel;

nrcSm151 = neutron.amount*k/eta*FgSm151;

PrcSm151 = nrcSm151*Sm151.A/NA;

CrcSm151 = nrcSm151*Sm151.A/NA;

Sm152.amount += 1.00*PrcSm151-1.00*CrcSm152+
1.00*PbndPm152/NA*Pm152.A;
```

```
PbndSm151 = ld*Sm151.halflife*Sm151.amount*NA/Sm151.A*cycle_time;
CbndSm151 = ld*Sm151.halflife*Sm151.amount*NA/Sm151.A*cycle_time;
Eu151.amount += 1.00*PbndSm151/NA*Sm151.A;
FgSm152 = Sm152.amount*Sm152.sigma_g/Sigma_a_fuel;
nrcSm152 = neutron.amount*k/eta*FgSm152;
PrcSm152 = nrcSm152*Sm152.A/NA;
CrcSm152 = nrcSm152*Sm152.A/NA;
Sm153.amount += 1.00*PrcSm152-1.00*CrcSm153+
1.00*PbndPm153/NA*Pm153.A;
FgSm153 = Sm153.amount*Sm153.sigma_g/Sigma_a_fuel;
nrcSm153 = neutron.amount*k/eta*FgSm153;
PrcSm153 = nrcSm153*Sm153.A/NA;
CrcSm153 = nrcSm153*Sm153.A/NA;
PbndSm153 = ld*Sm153.halflife*Sm153.amount*NA/Sm153.A*cycle_time;
CbndSm153 = ld*Sm153.halflife*Sm153.amount*NA/Sm153.A*cycle_time;
FgSm154 = Sm154.amount*Sm154.sigma_g/Sigma_a_fuel;
nrcSm154 = neutron.amount*k/eta*FgSm154;
PrcSm154 = nrcSm154*Sm154.A/NA;
CrcSm154 = nrcSm154*Sm154.A/NA;
Sm155.amount += 1.00*PrcSm154+1.00*PbndPm155/NA*Pm155.A;
PbndSm155 = ld*Sm155.halflife*Sm155.amount*NA/Sm155.A*cycle_time;
CbndSm155 = ld*Sm155.halflife*Sm155.amount*NA/Sm155.A*cycle_time;
PbndSm156 = ld*Sm156.halflife*Sm156.amount*NA/Sm156.A*cycle_time;
CbndSm156 = ld*Sm156.halflife*Sm156.amount*NA/Sm156.A*cycle_time;
PbndSm157 = ld*Sm157.halflife*Sm157.amount*NA/Sm157.A*cycle_time;
CbndSm157 = ld*Sm157.halflife*Sm157.amount*NA/Sm157.A*cycle_time;
Eu157.amount += 1.00*PbndSm157/NA*Sm157.A;
PbndSm158 = ld*Sm158.halflife*Sm158.amount*NA/Sm158.A*cycle_time;
CbndSm158 = ld*Sm158.halflife*Sm158.amount*NA/Sm158.A*cycle_time;
Eu158.amount += 1.00*PbndSm158/NA*Sm158.A;
PbndSm159 = ld*Sm159.halflife*Sm159.amount*NA/Sm159.A*cycle_time;
CbndSm159 = ld*Sm159.halflife*Sm159.amount*NA/Sm159.A*cycle_time;
```

```
Eu159.amount += 1.00*PbndSm159/NA*Sm159.A;
FgPm148 = Pm148.amount*Pm148.sigma_g/Sigma_a_fuel;
nrcPm148 = neutron.amount*k/eta*FgPm148;
PrcPm148 = nrcPm148*Pm148.A/NA;
CrcPm148 = nrcPm148*Pm148.A/NA;
Pm148.amount += -1.00*CrcPm148;
Pm149.amount += 1.00*PrcPm148+1.00*PbndNd149/NA*Nd149.A;
PbndPm148 = ld*Pm148.halflife*Pm148.amount*NA/Pm148.A*cycle_time;
CbndPm148 = ld*Pm148.halflife*Pm148.amount*NA/Pm148.A*cycle_time;
Sm148.amount += 1.00*PbndPm148/NA*Pm148.A;
PbndPm149 = ld*Pm149.halflife*Pm149.amount*NA/Pm149.A*cycle_time;
CbndPm149 = ld*Pm149.halflife*Pm149.amount*NA/Pm149.A*cycle_time;
Sm149.amount += 1.00*PbndPm149/NA*Pm149.A;
PbndPm150 = ld*Pm150.halflife*Pm150.amount*NA/Pm150.A*cycle_time;
CbndPm150 = ld*Pm150.halflife*Pm150.amount*NA/Pm150.A*cycle_time;
PbndPm151 = ld*Pm151.halflife*Pm151.amount*NA/Pm151.A*cycle_time;
CbndPm151 = ld*Pm151.halflife*Pm151.amount*NA/Pm151.A*cycle_time;
PbndPm152 = ld*Pm152.halflife*Pm152.amount*NA/Pm152.A*cycle_time;
CbndPm152 = ld*Pm152.halflife*Pm152.amount*NA/Pm152.A*cycle_time;
PbndPm153 = ld*Pm153.halflife*Pm153.amount*NA/Pm153.A*cycle_time;
CbndPm153 = ld*Pm153.halflife*Pm153.amount*NA/Pm153.A*cycle_time;
PbndPm154 = ld*Pm154.halflife*Pm154.amount*NA/Pm154.A*cycle_time;
CbndPm154 = ld*Pm154.halflife*Pm154.amount*NA/Pm154.A*cycle_time;
PbndPm155 = ld*Pm155.halflife*Pm155.amount*NA/Pm155.A*cycle_time;
CbndPm155 = ld*Pm155.halflife*Pm155.amount*NA/Pm155.A*cycle_time;
PbndPm156 = ld*Pm156.halflife*Pm156.amount*NA/Pm156.A*cycle_time;
CbndPm156 = ld*Pm156.halflife*Pm156.amount*NA/Pm156.A*cycle_time;
Sm156.amount += 1.00*PbndPm156/NA*Pm156.A;
PbndPm157 = ld*Pm157.halflife*Pm157.amount*NA/Pm157.A*cycle_time;
CbndPm157 = ld*Pm157.halflife*Pm157.amount*NA/Pm157.A*cycle_time;
Sm157.amount += 1.00*PbndPm157/NA*Pm157.A;
PbndPm158 = ld*Pm158.halflife*Pm158.amount*NA/Pm158.A*cycle_time;
```

```
CbndPm158 = ld*Pm158.halflife*Pm158.amount*NA/Pm158.A*cycle_time;
Sm158.amount += 1.00*PbndPm158/NA*Pm158.A;
FgNd145 = Nd145.amount*Nd145.sigma_g/Sigma_a_fuel;
nrcNd145 = neutron.amount*k/eta*FgNd145;
PrcNd145 = nrcNd145*Nd145.A/NA;
CrcNd145 = nrcNd145*Nd145.A/NA;
Nd145.amount += -1.00*CrcNd145+1.00*PbndPr145/NA*Pr145.A;
Nd146.amount += 1.00*PrcNd145-1.00*CrcNd146+
1.00*PbndPr146/NA*Pr146.A;
FgNd146 = Nd146.amount*Nd146.sigma_g/Sigma_a_fuel;
nrcNd146 = neutron.amount*k/eta*FgNd146;
PrcNd146 = nrcNd146*Nd146.A/NA;
CrcNd146 = nrcNd146*Nd146.A/NA;
Nd147.amount += 1.00*PrcNd146+1.00*PbndPr147/NA*Pr147.A;
PbndNd147 = ld*Nd147.halflife*Nd147.amount*NA/Nd147.A*cycle_time;
CbndNd147 = ld*Nd147.halflife*Nd147.amount*NA/Nd147.A*cycle_time;
Pm147.amount += 1.00*PbndNd147/NA*Nd147.A;
FgNd148 = Nd148.amount*Nd148.sigma_g/Sigma_a_fuel;
nrcNd148 = neutron.amount*k/eta*FgNd148;
PrcNd148 = nrcNd148*Nd148.A/NA;
CrcNd148 = nrcNd148*Nd148.A/NA;
Nd148.amount += -1.00*CrcNd148+1.00*PbndPr148/NA*Pr148.A;
Nd149.amount += 1.00*PrcNd148+1.00*PbndPr149/NA*Pr149.A;
PbndNd149 = ld*Nd149.halflife*Nd149.amount*NA/Nd149.A*cycle_time;
CbndNd149 = ld*Nd149.halflife*Nd149.amount*NA/Nd149.A*cycle_time;
FgNd150 = Nd150.amount*Nd150.sigma_g/Sigma_a_fuel;
nrcNd150 = neutron.amount*k/eta*FgNd150;
PrcNd150 = nrcNd150*Nd150.A/NA;
CrcNd150 = nrcNd150*Nd150.A/NA;
Nd150.amount += -1.00*CrcNd150+1.00*PbndPr150/NA*Pr150.A;
Nd151.amount += 1.00*PrcNd150+1.00*PbndPr151/NA*Pr151.A;
PbndNd151 = ld*Nd151.halflife*Nd151.amount*NA/Nd151.A*cycle_time;
CbndNd151 = ld*Nd151.halflife*Nd151.amount*NA/Nd151.A*cycle_time;
```

```
Pm151.amount += 1.00*PbndNd151/NA*Nd151.A;
PbndNd152 = ld*Nd152.halflife*Nd152.amount*NA/Nd152.A*cycle_time;
CbndNd152 = ld*Nd152.halflife*Nd152.amount*NA/Nd152.A*cycle_time;
Pm152.amount += 1.00*PbndNd152/NA*Nd152.A;
PbndNd153 = ld*Nd153.halflife*Nd153.amount*NA/Nd153.A*cycle_time;
CbndNd153 = ld*Nd153.halflife*Nd153.amount*NA/Nd153.A*cycle_time;
Pm153.amount += 1.00*PbndNd153/NA*Nd153.A;
PbndNd154 = ld*Nd154.halflife*Nd154.amount*NA/Nd154.A*cycle_time;
CbndNd154 = ld*Nd154.halflife*Nd154.amount*NA/Nd154.A*cycle_time;
Pm154.amount += 1.00*PbndNd154/NA*Nd154.A;
PbndNd155 = ld*Nd155.halflife*Nd155.amount*NA/Nd155.A*cycle_time;
CbndNd155 = ld*Nd155.halflife*Nd155.amount*NA/Nd155.A*cycle_time;
Pm155.amount += 1.00*PbndNd155/NA*Nd155.A;
PbndNd156 = ld*Nd156.halflife*Nd156.amount*NA/Nd156.A*cycle_time;
CbndNd156 = ld*Nd156.halflife*Nd156.amount*NA/Nd156.A*cycle_time;
Pm156.amount += 1.00*PbndNd156/NA*Nd156.A;
FgPr143 = Pr143.amount*Pr143.sigma_g/Sigma_a_fuel;
nrcPr143 = neutron.amount*k/eta*FgPr143;
PrcPr143 = nrcPr143*Pr143.A/NA;
CrcPr143 = nrcPr143*Pr143.A/NA;
Pr143.amount += -1.00*CrcPr143+1.00*PbndCe143/NA*Ce143.A;
Pr144.amount += 1.00*PrcPr143+1.00*PbndCe144/NA*Ce144.A;
PbndPr143 = ld*Pr143.halflife*Pr143.amount*NA/Pr143.A*cycle_time;
CbndPr143 = ld*Pr143.halflife*Pr143.amount*NA/Pr143.A*cycle_time;
Nd143.amount += 1.00*PbndPr143/NA*Pr143.A;
PbndPr144 = ld*Pr144.halflife*Pr144.amount*NA/Pr144.A*cycle_time;
CbndPr144 = ld*Pr144.halflife*Pr144.amount*NA/Pr144.A*cycle_time;
Nd144.amount += 1.00*PbndPr144/NA*Pr144.A;
PbndPr145 = ld*Pr145.halflife*Pr145.amount*NA/Pr145.A*cycle_time;
CbndPr145 = ld*Pr145.halflife*Pr145.amount*NA/Pr145.A*cycle_time;
PbndPr146 = ld*Pr146.halflife*Pr146.amount*NA/Pr146.A*cycle_time;
CbndPr146 = ld*Pr146.halflife*Pr146.amount*NA/Pr146.A*cycle_time;
```

```
PbndPr147 = ld*Pr147.halflife*Pr147.amount*NA/Pr147.A*cycle_time;
CbndPr147 = ld*Pr147.halflife*Pr147.amount*NA/Pr147.A*cycle_time;
PbndPr148 = ld*Pr148.halflife*Pr148.amount*NA/Pr148.A*cycle_time;
CbndPr148 = ld*Pr148.halflife*Pr148.amount*NA/Pr148.A*cycle_time;
PbndPr149 = ld*Pr149.halflife*Pr149.amount*NA/Pr149.A*cycle_time;
CbndPr149 = ld*Pr149.halflife*Pr149.amount*NA/Pr149.A*cycle_time;
PbndPr150 = ld*Pr150.halflife*Pr150.amount*NA/Pr150.A*cycle_time;
CbndPr150 = ld*Pr150.halflife*Pr150.amount*NA/Pr150.A*cycle_time;
PbndPr151 = ld*Pr151.halflife*Pr151.amount*NA/Pr151.A*cycle_time;
CbndPr151 = ld*Pr151.halflife*Pr151.amount*NA/Pr151.A*cycle_time;
PbndPr152 = ld*Pr152.halflife*Pr152.amount*NA/Pr152.A*cycle_time;
CbndPr152 = ld*Pr152.halflife*Pr152.amount*NA/Pr152.A*cycle_time;
Nd152.amount += 1.00*PbndPr152/NA*Pr152.A;
PbndPr153 = ld*Pr153.halflife*Pr153.amount*NA/Pr153.A*cycle_time;
CbndPr153 = ld*Pr153.halflife*Pr153.amount*NA/Pr153.A*cycle_time;
Nd153.amount += 1.00*PbndPr153/NA*Pr153.A;
PbndPr154 = ld*Pr154.halflife*Pr154.amount*NA/Pr154.A*cycle_time;
CbndPr154 = ld*Pr154.halflife*Pr154.amount*NA/Pr154.A*cycle_time;
Nd154.amount += 1.00*PbndPr154/NA*Pr154.A;
PbndCe141 = ld*Ce141.halflife*Ce141.amount*NA/Ce141.A*cycle_time;
CbndCe141 = ld*Ce141.halflife*Ce141.amount*NA/Ce141.A*cycle_time;
Pr141.amount += 1.00*PbndCe141/NA*Ce141.A;
PbndCe143 = ld*Ce143.halflife*Ce143.amount*NA/Ce143.A*cycle_time;
CbndCe143 = ld*Ce143.halflife*Ce143.amount*NA/Ce143.A*cycle_time;
PbndCe144 = ld*Ce144.halflife*Ce144.amount*NA/Ce144.A*cycle_time;
CbndCe144 = ld*Ce144.halflife*Ce144.amount*NA/Ce144.A*cycle_time;
PbndCe145 = ld*Ce145.halflife*Ce145.amount*NA/Ce145.A*cycle_time;
CbndCe145 = ld*Ce145.halflife*Ce145.amount*NA/Ce145.A*cycle_time;
Pr145.amount += 1.00*PbndCe145/NA*Ce145.A;
PbndCe146 = ld*Ce146.halflife*Ce146.amount*NA/Ce146.A*cycle_time;
CbndCe146 = ld*Ce146.halflife*Ce146.amount*NA/Ce146.A*cycle_time;
Pr146.amount += 1.00*PbndCe146/NA*Ce146.A;
```

```
PbndCe147 = ld*Ce147.halflife*Ce147.amount*NA/Ce147.A*cycle_time;
CbndCe147 = ld*Ce147.halflife*Ce147.amount*NA/Ce147.A*cycle_time;
Pr147.amount += 1.00*PbndCe147/NA*Ce147.A;
PbndCe148 = ld*Ce148.halflife*Ce148.amount*NA/Ce148.A*cycle_time;
CbndCe148 = ld*Ce148.halflife*Ce148.amount*NA/Ce148.A*cycle_time;
Pr148.amount += 1.00*PbndCe148/NA*Ce148.A;
PbndCe149 = ld*Ce149.halflife*Ce149.amount*NA/Ce149.A*cycle_time;
CbndCe149 = ld*Ce149.halflife*Ce149.amount*NA/Ce149.A*cycle_time;
Pr149.amount += 1.00*PbndCe149/NA*Ce149.A;
PbndCe150 = ld*Ce150.halflife*Ce150.amount*NA/Ce150.A*cycle_time;
CbndCe150 = ld*Ce150.halflife*Ce150.amount*NA/Ce150.A*cycle_time;
Pr150.amount += 1.00*PbndCe150/NA*Ce150.A;
PbndCe151 = ld*Ce151.halflife*Ce151.amount*NA/Ce151.A*cycle_time;
CbndCe151 = ld*Ce151.halflife*Ce151.amount*NA/Ce151.A*cycle_time;
Pr151.amount += 1.00*PbndCe151/NA*Ce151.A;
PbndCe152 = ld*Ce152.halflife*Ce152.amount*NA/Ce152.A*cycle_time;
CbndCe152 = ld*Ce152.halflife*Ce152.amount*NA/Ce152.A*cycle_time;
Pr152.amount += 1.00*PbndCe152/NA*Ce152.A;
FgLa138 = La138.amount*La138.sigma_g/Sigma_a_fuel;
nrcLa138 = neutron.amount*k/eta*FgLa138;
PrcLa138 = nrcLa138*La138.A/NA;
CrcLa138 = nrcLa138*La138.A/NA;
La138.amount += -1.00*CrcLa138;
La139.amount += 1.00*PrcLa138+1.00*PbndBa139/NA*Ba139.A;
PbndLa140 = ld*La140.halflife*La140.amount*NA/La140.A*cycle_time;
CbndLa140 = ld*La140.halflife*La140.amount*NA/La140.A*cycle_time;
Ce140.amount += 1.00*PbndLa140/NA*La140.A;
PbndLa141 = ld*La141.halflife*La141.amount*NA/La141.A*cycle_time;
CbndLa141 = ld*La141.halflife*La141.amount*NA/La141.A*cycle_time;
Ce141.amount += 1.00*PbndLa141/NA*La141.A;
PbndLa142 = ld*La142.halflife*La142.amount*NA/La142.A*cycle_time;
CbndLa142 = ld*La142.halflife*La142.amount*NA/La142.A*cycle_time;
```

```
Ce142.amount += 1.00*PbndLa142/NA*La142.A;
PbndLa143 = ld*La143.halflife*La143.amount*NA/La143.A*cycle_time;
CbndLa143 = ld*La143.halflife*La143.amount*NA/La143.A*cycle_time;
Ce143.amount += 1.00*PbndLa143/NA*La143.A;
PbndLa144 = ld*La144.halflife*La144.amount*NA/La144.A*cycle_time;
CbndLa144 = ld*La144.halflife*La144.amount*NA/La144.A*cycle_time;
Ce144.amount += 1.00*PbndLa144/NA*La144.A;
PbndLa145 = ld*La145.halflife*La145.amount*NA/La145.A*cycle_time;
CbndLa145 = ld*La145.halflife*La145.amount*NA/La145.A*cycle_time;
Ce145.amount += 1.00*PbndLa145/NA*La145.A;
PbndLa146 = ld*La146.halflife*La146.amount*NA/La146.A*cycle_time;
CbndLa146 = ld*La146.halflife*La146.amount*NA/La146.A*cycle_time;
Ce146.amount += 1.00*PbndLa146/NA*La146.A;
PbndLa147 = ld*La147.halflife*La147.amount*NA/La147.A*cycle_time;
CbndLa147 = ld*La147.halflife*La147.amount*NA/La147.A*cycle_time;
Ce147.amount += 1.00*PbndLa147/NA*La147.A;
PbndLa148 = ld*La148.halflife*La148.amount*NA/La148.A*cycle_time;
CbndLa148 = ld*La148.halflife*La148.amount*NA/La148.A*cycle_time;
Ce148.amount += 1.00*PbndLa148/NA*La148.A;
PbndLa149 = ld*La149.halflife*La149.amount*NA/La149.A*cycle_time;
CbndLa149 = ld*La149.halflife*La149.amount*NA/La149.A*cycle_time;
Ce149.amount += 1.00*PbndLa149/NA*La149.A;
PbndBa139 = ld*Ba139.halflife*Ba139.amount*NA/Ba139.A*cycle_time;
CbndBa139 = ld*Ba139.halflife*Ba139.amount*NA/Ba139.A*cycle_time;
PbndBa140 = ld*Ba140.halflife*Ba140.amount*NA/Ba140.A*cycle_time;
CbndBa140 = ld*Ba140.halflife*Ba140.amount*NA/Ba140.A*cycle_time;
La140.amount += 1.00*PbndBa140/NA*Ba140.A;
PbndBa141 = ld*Ba141.halflife*Ba141.amount*NA/Ba141.A*cycle_time;
CbndBa141 = ld*Ba141.halflife*Ba141.amount*NA/Ba141.A*cycle_time;
La141.amount += 1.00*PbndBa141/NA*Ba141.A;
PbndBa142 = ld*Ba142.halflife*Ba142.amount*NA/Ba142.A*cycle_time;
CbndBa142 = ld*Ba142.halflife*Ba142.amount*NA/Ba142.A*cycle_time;
```

```
La142.amount += 1.00*PbndBa142/NA*Ba142.A;
PbndBa143 = ld*Ba143.halflife*Ba143.amount*NA/Ba143.A*cycle_time;
CbndBa143 = ld*Ba143.halflife*Ba143.amount*NA/Ba143.A*cycle_time;
La143.amount += 1.00*PbndBa143/NA*Ba143.A;
PbndBa144 = ld*Ba144.halflife*Ba144.amount*NA/Ba144.A*cycle_time;
CbndBa144 = ld*Ba144.halflife*Ba144.amount*NA/Ba144.A*cycle_time;
La144.amount += 1.00*PbndBa144/NA*Ba144.A;
PbndBa145 = ld*Ba145.halflife*Ba145.amount*NA/Ba145.A*cycle_time;
CbndBa145 = ld*Ba145.halflife*Ba145.amount*NA/Ba145.A*cycle_time;
La145.amount += 1.00*PbndBa145/NA*Ba145.A;
PbndBa146 = ld*Ba146.halflife*Ba146.amount*NA/Ba146.A*cycle_time;
CbndBa146 = ld*Ba146.halflife*Ba146.amount*NA/Ba146.A*cycle_time;
La146.amount += 1.00*PbndBa146/NA*Ba146.A;
PbndBa147 = ld*Ba147.halflife*Ba147.amount*NA/Ba147.A*cycle_time;
CbndBa147 = ld*Ba147.halflife*Ba147.amount*NA/Ba147.A*cycle_time;
La147.amount += 1.00*PbndBa147/NA*Ba147.A;
PbndBa148 = ld*Ba148.halflife*Ba148.amount*NA/Ba148.A*cycle_time;
CbndBa148 = ld*Ba148.halflife*Ba148.amount*NA/Ba148.A*cycle_time;
La148.amount += 1.00*PbndBa148/NA*Ba148.A;
PbndBa149 = ld*Ba149.halflife*Ba149.amount*NA/Ba149.A*cycle_time;
CbndBa149 = ld*Ba149.halflife*Ba149.amount*NA/Ba149.A*cycle_time;
La149.amount += 1.00*PbndBa149/NA*Ba149.A;
FgCs133 = Cs133.amount*Cs133.sigma_g/Sigma_a_fuel;
nrcCs133 = neutron.amount*k/eta*FgCs133;
PrcCs133 = nrcCs133*Cs133.A/NA;
CrcCs133 = nrcCs133*Cs133.A/NA;
Cs133.amount += -1.00*CrcCs133+1.00*PbndXe133/NA*Xe133.A;
Cs134.amount += 1.00*PrcCs133;
PbndCs134 = ld*Cs134.halflife*Cs134.amount*NA/Cs134.A*cycle_time;
CbndCs134 = ld*Cs134.halflife*Cs134.amount*NA/Cs134.A*cycle_time;
Ba134.amount += 1.00*PbndCs134/NA*Cs134.A;
PbndCs136 = ld*Cs136.halflife*Cs136.amount*NA/Cs136.A*cycle_time;
```

```
CbndCs136 = ld*Cs136.halflife*Cs136.amount*NA/Cs136.A*cycle_time;
Ba136.amount += 1.00*PbndCs136/NA*Cs136.A;
PbndCs137 = ld*Cs137.halflife*Cs137.amount*NA/Cs137.A*cycle_time;
CbndCs137 = ld*Cs137.halflife*Cs137.amount*NA/Cs137.A*cycle_time;
Ba137.amount += 1.00*PbndCs137/NA*Cs137.A;
PbndCs138 = ld*Cs138.halflife*Cs138.amount*NA/Cs138.A*cycle_time;
CbndCs138 = ld*Cs138.halflife*Cs138.amount*NA/Cs138.A*cycle_time;
Ba138.amount += 1.00*PbndCs138/NA*Cs138.A;
PbndCs139 = ld*Cs139.halflife*Cs139.amount*NA/Cs139.A*cycle_time;
CbndCs139 = ld*Cs139.halflife*Cs139.amount*NA/Cs139.A*cycle_time;
Ba139.amount += 1.00*PbndCs139/NA*Cs139.A;
PbndCs140 = ld*Cs140.halflife*Cs140.amount*NA/Cs140.A*cycle_time;
CbndCs140 = ld*Cs140.halflife*Cs140.amount*NA/Cs140.A*cycle_time;
Ba140.amount += 1.00*PbndCs140/NA*Cs140.A;
PbndCs141 = ld*Cs141.halflife*Cs141.amount*NA/Cs141.A*cycle_time;
CbndCs141 = ld*Cs141.halflife*Cs141.amount*NA/Cs141.A*cycle_time;
Ba141.amount += 1.00*PbndCs141/NA*Cs141.A;
PbndCs142 = ld*Cs142.halflife*Cs142.amount*NA/Cs142.A*cycle_time;
CbndCs142 = ld*Cs142.halflife*Cs142.amount*NA/Cs142.A*cycle_time;
Ba142.amount += 1.00*PbndCs142/NA*Cs142.A;
PbndCs143 = ld*Cs143.halflife*Cs143.amount*NA/Cs143.A*cycle_time;
CbndCs143 = ld*Cs143.halflife*Cs143.amount*NA/Cs143.A*cycle_time;
Ba143.amount += 1.00*PbndCs143/NA*Cs143.A;
PbndCs144 = ld*Cs144.halflife*Cs144.amount*NA/Cs144.A*cycle_time;
CbndCs144 = ld*Cs144.halflife*Cs144.amount*NA/Cs144.A*cycle_time;
Ba144.amount += 1.00*PbndCs144/NA*Cs144.A;
PbndCs145 = ld*Cs145.halflife*Cs145.amount*NA/Cs145.A*cycle_time;
CbndCs145 = ld*Cs145.halflife*Cs145.amount*NA/Cs145.A*cycle_time;
Ba145.amount += 1.00*PbndCs145/NA*Cs145.A;
PbndCs146 = ld*Cs146.halflife*Cs146.amount*NA/Cs146.A*cycle_time;
CbndCs146 = ld*Cs146.halflife*Cs146.amount*NA/Cs146.A*cycle_time;
Ba146.amount += 1.00*PbndCs146/NA*Cs146.A;
```

```
PbndCs147 = ld*Cs147.halflife*Cs147.amount*NA/Cs147.A*cycle_time;
CbndCs147 = ld*Cs147.halflife*Cs147.amount*NA/Cs147.A*cycle_time;
Ba147.amount += 1.00*PbndCs147/NA*Cs147.A;
PbndCs148 = ld*Cs148.halflife*Cs148.amount*NA/Cs148.A*cycle_time;
CbndCs148 = ld*Cs148.halflife*Cs148.amount*NA/Cs148.A*cycle_time;
Ba148.amount += 1.00*PbndCs148/NA*Cs148.A;
FgXe130 = Xe130.amount*Xe130.sigma_g/Sigma_a_fuel;
nrcXe130 = neutron.amount*k/eta*FgXe130;
PrcXe130 = nrcXe130*Xe130.A/NA;
CrcXe130 = nrcXe130*Xe130.A/NA;
Xe130.amount += -1.00*CrcXe130+1.00*PbndI130/NA*I130.A;
Xe131.amount += 1.00*PrcXe130-1.00*CrcXe131+1.00*PbndI131/NA*I131.A;
FgXe131 = Xe131.amount*Xe131.sigma_g/Sigma_a_fuel;
nrcXe131 = neutron.amount*k/eta*FgXe131;
PrcXe131 = nrcXe131*Xe131.A/NA;
CrcXe131 = nrcXe131*Xe131.A/NA;
Xe132.amount += 1.00*PrcXe131+1.00*PbndI132/NA*I132.A;
PbndXe133 = ld*Xe133.halflife*Xe133.amount*NA/Xe133.A*cycle_time;
CbndXe133 = ld*Xe133.halflife*Xe133.amount*NA/Xe133.A*cycle_time;
FgXe135 = Xe135.amount*Xe135.sigma_g/Sigma_a_fuel;
nrcXe135 = neutron.amount*k/eta*FgXe135;
PrcXe135 = nrcXe135*Xe135.A/NA;
CrcXe135 = nrcXe135*Xe135.A/NA;
Xe135.amount += -1.00*CrcXe135+1.00*PbndI135/NA*I135.A;
Xe136.amount += 1.00*PrcXe135+1.00*PbndI136/NA*I136.A;
PbndXe135 = ld*Xe135.halflife*Xe135.amount*NA/Xe135.A*cycle_time;
CbndXe135 = ld*Xe135.halflife*Xe135.amount*NA/Xe135.A*cycle_time;
Cs135.amount += 1.00*PbndXe135/NA*Xe135.A;
PbndXe137 = ld*Xe137.halflife*Xe137.amount*NA/Xe137.A*cycle_time;
CbndXe137 = ld*Xe137.halflife*Xe137.amount*NA/Xe137.A*cycle_time;
Cs137.amount += 1.00*PbndXe137/NA*Xe137.A;
PbndXe138 = ld*Xe138.halflife*Xe138.amount*NA/Xe138.A*cycle_time;
```

```
CbndXe138 = ld*Xe138.halflife*Xe138.amount*NA/Xe138.A*cycle_time;
Cs138.amount += 1.00*PbndXe138/NA*Xe138.A;
PbndXe139 = ld*Xe139.halflife*Xe139.amount*NA/Xe139.A*cycle_time;
CbndXe139 = ld*Xe139.halflife*Xe139.amount*NA/Xe139.A*cycle_time;
Cs139.amount += 1.00*PbndXe139/NA*Xe139.A;
PbndXe140 = ld*Xe140.halflife*Xe140.amount*NA/Xe140.A*cycle_time;
CbndXe140 = ld*Xe140.halflife*Xe140.amount*NA/Xe140.A*cycle_time;
Cs140.amount += 1.00*PbndXe140/NA*Xe140.A;
PbndXe141 = ld*Xe141.halflife*Xe141.amount*NA/Xe141.A*cycle_time;
CbndXe141 = ld*Xe141.halflife*Xe141.amount*NA/Xe141.A*cycle_time;
Cs141.amount += 1.00*PbndXe141/NA*Xe141.A;
PbndXe142 = ld*Xe142.halflife*Xe142.amount*NA/Xe142.A*cycle_time;
CbndXe142 = ld*Xe142.halflife*Xe142.amount*NA/Xe142.A*cycle_time;
Cs142.amount += 1.00*PbndXe142/NA*Xe142.A;
PbndXe143 = ld*Xe143.halflife*Xe143.amount*NA/Xe143.A*cycle_time;
CbndXe143 = ld*Xe143.halflife*Xe143.amount*NA/Xe143.A*cycle_time;
Cs143.amount += 1.00*PbndXe143/NA*Xe143.A;
PbndXe144 = ld*Xe144.halflife*Xe144.amount*NA/Xe144.A*cycle_time;
CbndXe144 = ld*Xe144.halflife*Xe144.amount*NA/Xe144.A*cycle_time;
Cs144.amount += 1.00*PbndXe144/NA*Xe144.A;
PbndXe145 = ld*Xe145.halflife*Xe145.amount*NA/Xe145.A*cycle_time;
CbndXe145 = ld*Xe145.halflife*Xe145.amount*NA/Xe145.A*cycle_time;
Cs145.amount += 1.00*PbndXe145/NA*Xe145.A;
FgI127 = I127.amount*I127.sigma_g/Sigma_a_fuel;
nrcI127 = neutron.amount*k/eta*FgI127;
PrcI127 = nrcI127*I127.A/NA;
CrcI127 = nrcI127*I127.A/NA;
I127.amount += -1.00*CrcI127+1.00*PbndTe127/NA*Te127.A;
I128.amount += 1.00*PrcI127;
PbndI128 = ld*I128.halflife*I128.amount*NA/I128.A*cycle_time;
CbndI128 = ld*I128.halflife*I128.amount*NA/I128.A*cycle_time;
Xe128.amount += 1.00*PbndI128/NA*I128.A;
```

```
FgI129 = I129.amount*I129.sigma_g/Sigma_a_fuel;
nrcI129 = neutron.amount*k/eta*FgI129;
PrcI129 = nrcI129*I129.A/NA;
CrcI129 = nrcI129*I129.A/NA;
I129.amount += -1.00*CrcI129+1.00*PbndTe129/NA*Te129.A;
I130.amount += 1.00*PrcI129;
PbndI130 = ld*I130.halflife*I130.amount*NA/I130.A*cycle_time;
CbndI130 = ld*I130.halflife*I130.amount*NA/I130.A*cycle_time;
PbndI131 = ld*I131.halflife*I131.amount*NA/I131.A*cycle_time;
CbndI131 = ld*I131.halflife*I131.amount*NA/I131.A*cycle_time;
PbndI132 = ld*I132.halflife*I132.amount*NA/I132.A*cycle_time;
CbndI132 = ld*I132.halflife*I132.amount*NA/I132.A*cycle_time;
PbndI133 = ld*I133.halflife*I133.amount*NA/I133.A*cycle_time;
CbndI133 = ld*I133.halflife*I133.amount*NA/I133.A*cycle_time;
Xe133.amount += 1.00*PbndI133/NA*I133.A;
PbndI134 = ld*I134.halflife*I134.amount*NA/I134.A*cycle_time;
CbndI134 = ld*I134.halflife*I134.amount*NA/I134.A*cycle_time;
Xe134.amount += 1.00*PbndI134/NA*I134.A;
PbndI135 = ld*I135.halflife*I135.amount*NA/I135.A*cycle_time;
CbndI135 = ld*I135.halflife*I135.amount*NA/I135.A*cycle_time;
PbndI136 = ld*I136.halflife*I136.amount*NA/I136.A*cycle_time;
CbndI136 = ld*I136.halflife*I136.amount*NA/I136.A*cycle_time;
PbndI137 = ld*I137.halflife*I137.amount*NA/I137.A*cycle_time;
CbndI137 = ld*I137.halflife*I137.amount*NA/I137.A*cycle_time;
Xe137.amount += 1.00*PbndI137/NA*I137.A;
PbndI138 = ld*I138.halflife*I138.amount*NA/I138.A*cycle_time;
CbndI138 = ld*I138.halflife*I138.amount*NA/I138.A*cycle_time;
Xe138.amount += 1.00*PbndI138/NA*I138.A;
PbndI139 = ld*I139.halflife*I139.amount*NA/I139.A*cycle_time;
CbndI139 = ld*I139.halflife*I139.amount*NA/I139.A*cycle_time;
Xe139.amount += 1.00*PbndI139/NA*I139.A;
PbndI140 = ld*I140.halflife*I140.amount*NA/I140.A*cycle_time;
```

```
CbndI140 = ld*I140.halflife*I140.amount*NA/I140.A*cycle_time;
Xe140.amount += 1.00*PbndI140/NA*I140.A;
PbndI141 = ld*I141.halflife*I141.amount*NA/I141.A*cycle_time;
CbndI141 = ld*I141.halflife*I141.amount*NA/I141.A*cycle_time;
Xe141.amount += 1.00*PbndI141/NA*I141.A;
PbndI142 = ld*I142.halflife*I142.amount*NA/I142.A*cycle_time;
CbndI142 = ld*I142.halflife*I142.amount*NA/I142.A*cycle_time;
Xe142.amount += 1.00*PbndI142/NA*I142.A;
PbndTe127 = ld*Te127.halflife*Te127.amount*NA/Te127.A*cycle_time;
CbndTe127 = ld*Te127.halflife*Te127.amount*NA/Te127.A*cycle_time;
PbndTe129 = ld*Te129.halflife*Te129.amount*NA/Te129.A*cycle_time;
CbndTe129 = ld*Te129.halflife*Te129.amount*NA/Te129.A*cycle_time;
PbndTe131 = ld*Te131.halflife*Te131.amount*NA/Te131.A*cycle_time;
CbndTe131 = ld*Te131.halflife*Te131.amount*NA/Te131.A*cycle_time;
I131.amount += 1.00*PbndTe131/NA*Te131.A;
PbndTe132 = ld*Te132.halflife*Te132.amount*NA/Te132.A*cycle_time;
CbndTe132 = ld*Te132.halflife*Te132.amount*NA/Te132.A*cycle_time;
I132.amount += 1.00*PbndTe132/NA*Te132.A;
PbndTe133 = ld*Te133.halflife*Te133.amount*NA/Te133.A*cycle_time;
CbndTe133 = ld*Te133.halflife*Te133.amount*NA/Te133.A*cycle_time;
I133.amount += 1.00*PbndTe133/NA*Te133.A;
PbndTe134 = ld*Te134.halflife*Te134.amount*NA/Te134.A*cycle_time;
CbndTe134 = ld*Te134.halflife*Te134.amount*NA/Te134.A*cycle_time;
I134.amount += 1.00*PbndTe134/NA*Te134.A;
PbndTe135 = ld*Te135.halflife*Te135.amount*NA/Te135.A*cycle_time;
CbndTe135 = ld*Te135.halflife*Te135.amount*NA/Te135.A*cycle_time;
I135.amount += 1.00*PbndTe135/NA*Te135.A;
PbndTe136 = ld*Te136.halflife*Te136.amount*NA/Te136.A*cycle_time;
CbndTe136 = ld*Te136.halflife*Te136.amount*NA/Te136.A*cycle_time;
I136.amount += 1.00*PbndTe136/NA*Te136.A;
PbndTe137 = ld*Te137.halflife*Te137.amount*NA/Te137.A*cycle_time;
CbndTe137 = ld*Te137.halflife*Te137.amount*NA/Te137.A*cycle_time;
```

```
I137.amount += 1.00*PbndTe137/NA*Te137.A;
PbndTe138 = ld*Te138.halflife*Te138.amount*NA/Te138.A*cycle_time;
CbndTe138 = ld*Te138.halflife*Te138.amount*NA/Te138.A*cycle_time;
I138.amount += 1.00*PbndTe138/NA*Te138.A;
PbndSb122 = ld*Sb122.halflife*Sb122.amount*NA/Sb122.A*cycle_time;
CbndSb122 = ld*Sb122.halflife*Sb122.amount*NA/Sb122.A*cycle_time;
Te122.amount += 1.00*PbndSb122/NA*Sb122.A;
FgSb123 = Sb123.amount*Sb123.sigma_g/Sigma_a_fuel;
nrcSb123 = neutron.amount*k/eta*FgSb123;
PrcSb123 = nrcSb123*Sb123.A/NA;
CrcSb123 = nrcSb123*Sb123.A/NA;
Sb123.amount += -1.00*CrcSb123+1.00*PbndSn123/NA*Sn123.A;
Sb124.amount += 1.00*PrcSb123;
PbndSb124 = ld*Sb124.halflife*Sb124.amount*NA/Sb124.A*cycle_time;
CbndSb124 = ld*Sb124.halflife*Sb124.amount*NA/Sb124.A*cycle_time;
Te124.amount += 1.00*PbndSb124/NA*Sb124.A;
PbndSb125 = ld*Sb125.halflife*Sb125.amount*NA/Sb125.A*cycle_time;
CbndSb125 = ld*Sb125.halflife*Sb125.amount*NA/Sb125.A*cycle_time;
Te125.amount += 1.00*PbndSb125/NA*Sb125.A;
PbndSb126 = ld*Sb126.halflife*Sb126.amount*NA/Sb126.A*cycle_time;
CbndSb126 = ld*Sb126.halflife*Sb126.amount*NA/Sb126.A*cycle_time;
Te126.amount += 1.00*PbndSb126/NA*Sb126.A;
PbndSb127 = ld*Sb127.halflife*Sb127.amount*NA/Sb127.A*cycle_time;
CbndSb127 = ld*Sb127.halflife*Sb127.amount*NA/Sb127.A*cycle_time;
Te127.amount += 1.00*PbndSb127/NA*Sb127.A;
PbndSb128 = ld*Sb128.halflife*Sb128.amount*NA/Sb128.A*cycle_time;
CbndSb128 = ld*Sb128.halflife*Sb128.amount*NA/Sb128.A*cycle_time;
Te128.amount += 1.00*PbndSb128/NA*Sb128.A;
PbndSb129 = ld*Sb129.halflife*Sb129.amount*NA/Sb129.A*cycle_time;
CbndSb129 = ld*Sb129.halflife*Sb129.amount*NA/Sb129.A*cycle_time;
Te129.amount += 1.00*PbndSb129/NA*Sb129.A;
PbndSb130 = ld*Sb130.halflife*Sb130.amount*NA/Sb130.A*cycle_time;
```

```
CbndSb130 = ld*Sb130.halflife*Sb130.amount*NA/Sb130.A*cycle_time;
Te130.amount += 1.00*PbndSb130/NA*Sb130.A;
PbndSb131 = ld*Sb131.halflife*Sb131.amount*NA/Sb131.A*cycle_time;
CbndSb131 = ld*Sb131.halflife*Sb131.amount*NA/Sb131.A*cycle_time;
Te131.amount += 1.00*PbndSb131/NA*Sb131.A;
PbndSb132 = ld*Sb132.halflife*Sb132.amount*NA/Sb132.A*cycle_time;
CbndSb132 = ld*Sb132.halflife*Sb132.amount*NA/Sb132.A*cycle_time;
Te132.amount += 1.00*PbndSb132/NA*Sb132.A;
PbndSb133 = ld*Sb133.halflife*Sb133.amount*NA/Sb133.A*cycle_time;
CbndSb133 = ld*Sb133.halflife*Sb133.amount*NA/Sb133.A*cycle_time;
Te133.amount += 1.00*PbndSb133/NA*Sb133.A;
PbndSb134 = ld*Sb134.halflife*Sb134.amount*NA/Sb134.A*cycle_time;
CbndSb134 = ld*Sb134.halflife*Sb134.amount*NA/Sb134.A*cycle_time;
Te134.amount += 1.00*PbndSb134/NA*Sb134.A;
PbndSb135 = ld*Sb135.halflife*Sb135.amount*NA/Sb135.A*cycle_time;
CbndSb135 = ld*Sb135.halflife*Sb135.amount*NA/Sb135.A*cycle_time;
Te135.amount += 1.00*PbndSb135/NA*Sb135.A;
PbndSb136 = ld*Sb136.halflife*Sb136.amount*NA/Sb136.A*cycle_time;
CbndSb136 = ld*Sb136.halflife*Sb136.amount*NA/Sb136.A*cycle_time;
Te136.amount += 1.00*PbndSb136/NA*Sb136.A;
PbndSn121 = ld*Sn121.halflife*Sn121.amount*NA/Sn121.A*cycle_time;
CbndSn121 = ld*Sn121.halflife*Sn121.amount*NA/Sn121.A*cycle_time;
Sb121.amount += 1.00*PbndSn121/NA*Sn121.A;
PbndSn123 = ld*Sn123.halflife*Sn123.amount*NA/Sn123.A*cycle_time;
CbndSn123 = ld*Sn123.halflife*Sn123.amount*NA/Sn123.A*cycle_time;
PbndSn125 = ld*Sn125.halflife*Sn125.amount*NA/Sn125.A*cycle_time;
CbndSn125 = ld*Sn125.halflife*Sn125.amount*NA/Sn125.A*cycle_time;
Sb125.amount += 1.00*PbndSn125/NA*Sn125.A;
PbndSn127 = ld*Sn127.halflife*Sn127.amount*NA/Sn127.A*cycle_time;
CbndSn127 = ld*Sn127.halflife*Sn127.amount*NA/Sn127.A*cycle_time;
Sb127.amount += 1.00*PbndSn127/NA*Sn127.A;
PbndSn128 = ld*Sn128.halflife*Sn128.amount*NA/Sn128.A*cycle_time;
```

```
CbndSn128 = ld*Sn128.halflife*Sn128.amount*NA/Sn128.A*cycle_time;
Sb128.amount += 1.00*PbndSn128/NA*Sn128.A;
PbndSn129 = ld*Sn129.halflife*Sn129.amount*NA/Sn129.A*cycle_time;
CbndSn129 = ld*Sn129.halflife*Sn129.amount*NA/Sn129.A*cycle_time;
Sb129.amount += 1.00*PbndSn129/NA*Sn129.A;
PbndSn130 = ld*Sn130.halflife*Sn130.amount*NA/Sn130.A*cycle_time;
CbndSn130 = ld*Sn130.halflife*Sn130.amount*NA/Sn130.A*cycle_time;
Sb130.amount += 1.00*PbndSn130/NA*Sn130.A;
PbndSn131 = ld*Sn131.halflife*Sn131.amount*NA/Sn131.A*cycle_time;
CbndSn131 = ld*Sn131.halflife*Sn131.amount*NA/Sn131.A*cycle_time;
Sb131.amount += 1.00*PbndSn131/NA*Sn131.A;
PbndSn132 = ld*Sn132.halflife*Sn132.amount*NA/Sn132.A*cycle_time;
CbndSn132 = ld*Sn132.halflife*Sn132.amount*NA/Sn132.A*cycle_time;
Sb132.amount += 1.00*PbndSn132/NA*Sn132.A;
PbndSn133 = ld*Sn133.halflife*Sn133.amount*NA/Sn133.A*cycle_time;
CbndSn133 = ld*Sn133.halflife*Sn133.amount*NA/Sn133.A*cycle_time;
Sb133.amount += 1.00*PbndSn133/NA*Sn133.A;
PbndSn134 = ld*Sn134.halflife*Sn134.amount*NA/Sn134.A*cycle_time;
CbndSn134 = ld*Sn134.halflife*Sn134.amount*NA/Sn134.A*cycle_time;
Sb134.amount += 1.00*PbndSn134/NA*Sn134.A;
PbndIn117 = ld*In117.halflife*In117.amount*NA/In117.A*cycle_time;
CbndIn117 = ld*In117.halflife*In117.amount*NA/In117.A*cycle_time;
Sn117.amount += 1.00*PbndIn117/NA*In117.A;
PbndIn118 = ld*In118.halflife*In118.amount*NA/In118.A*cycle_time;
CbndIn118 = ld*In118.halflife*In118.amount*NA/In118.A*cycle_time;
Sn118.amount += 1.00*PbndIn118/NA*In118.A;
PbndIn119 = ld*In119.halflife*In119.amount*NA/In119.A*cycle_time;
CbndIn119 = ld*In119.halflife*In119.amount*NA/In119.A*cycle_time;
Sn119.amount += 1.00*PbndIn119/NA*In119.A;
PbndIn120 = ld*In120.halflife*In120.amount*NA/In120.A*cycle_time;
CbndIn120 = ld*In120.halflife*In120.amount*NA/In120.A*cycle_time;
Sn120.amount += 1.00*PbndIn120/NA*In120.A;
```

```
PbndIn121 = ld*In121.halflife*In121.amount*NA/In121.A*cycle_time;
CbndIn121 = ld*In121.halflife*In121.amount*NA/In121.A*cycle_time;
Sn121.amount += 1.00*PbndIn121/NA*In121.A;
PbndIn122 = ld*In122.halflife*In122.amount*NA/In122.A*cycle_time;
CbndIn122 = ld*In122.halflife*In122.amount*NA/In122.A*cycle_time;
Sn122.amount += 1.00*PbndIn122/NA*In122.A;
PbndIn123 = ld*In123.halflife*In123.amount*NA/In123.A*cycle_time;
CbndIn123 = ld*In123.halflife*In123.amount*NA/In123.A*cycle_time;
Sn123.amount += 1.00*PbndIn123/NA*In123.A;
PbndIn124 = ld*In124.halflife*In124.amount*NA/In124.A*cycle_time;
CbndIn124 = ld*In124.halflife*In124.amount*NA/In124.A*cycle_time;
Sn124.amount += 1.00*PbndIn124/NA*In124.A;
PbndIn125 = ld*In125.halflife*In125.amount*NA/In125.A*cycle_time;
CbndIn125 = ld*In125.halflife*In125.amount*NA/In125.A*cycle_time;
Sn125.amount += 1.00*PbndIn125/NA*In125.A;
PbndIn126 = ld*In126.halflife*In126.amount*NA/In126.A*cycle_time;
CbndIn126 = ld*In126.halflife*In126.amount*NA/In126.A*cycle_time;
Sn126.amount += 1.00*PbndIn126/NA*In126.A;
PbndIn127 = ld*In127.halflife*In127.amount*NA/In127.A*cycle_time;
CbndIn127 = ld*In127.halflife*In127.amount*NA/In127.A*cycle_time;
Sn127.amount += 1.00*PbndIn127/NA*In127.A;
PbndIn128 = ld*In128.halflife*In128.amount*NA/In128.A*cycle_time;
CbndIn128 = ld*In128.halflife*In128.amount*NA/In128.A*cycle_time;
Sn128.amount += 1.00*PbndIn128/NA*In128.A;
PbndIn129 = ld*In129.halflife*In129.amount*NA/In129.A*cycle_time;
CbndIn129 = ld*In129.halflife*In129.amount*NA/In129.A*cycle_time;
Sn129.amount += 1.00*PbndIn129/NA*In129.A;
PbndIn130 = ld*In130.halflife*In130.amount*NA/In130.A*cycle_time;
CbndIn130 = ld*In130.halflife*In130.amount*NA/In130.A*cycle_time;
Sn130.amount += 1.00*PbndIn130/NA*In130.A;
PbndIn131 = ld*In131.halflife*In131.amount*NA/In131.A*cycle_time;
CbndIn131 = ld*In131.halflife*In131.amount*NA/In131.A*cycle_time;
```

```
Sn131.amount += 1.00*PbndIn131/NA*In131.A;
PbndIn132 = ld*In132.halflife*In132.amount*NA/In132.A*cycle_time;
CbndIn132 = ld*In132.halflife*In132.amount*NA/In132.A*cycle_time;
Sn132.amount += 1.00*PbndIn132/NA*In132.A;
PbndIn133 = ld*In133.halflife*In133.amount*NA/In133.A*cycle_time;
CbndIn133 = ld*In133.halflife*In133.amount*NA/In133.A*cycle_time;
Sn133.amount += 1.00*PbndIn133/NA*In133.A;
PbndCd115 = ld*Cd115.halflife*Cd115.amount*NA/Cd115.A*cycle_time;
CbndCd115 = ld*Cd115.halflife*Cd115.amount*NA/Cd115.A*cycle_time;
In115.amount += 1.00*PbndCd115/NA*Cd115.A;
PbndCd117 = ld*Cd117.halflife*Cd117.amount*NA/Cd117.A*cycle_time;
CbndCd117 = ld*Cd117.halflife*Cd117.amount*NA/Cd117.A*cycle_time;
In117.amount += 1.00*PbndCd117/NA*Cd117.A;
PbndCd118 = ld*Cd118.halflife*Cd118.amount*NA/Cd118.A*cycle_time;
CbndCd118 = ld*Cd118.halflife*Cd118.amount*NA/Cd118.A*cycle_time;
In118.amount += 1.00*PbndCd118/NA*Cd118.A;
PbndCd119 = ld*Cd119.halflife*Cd119.amount*NA/Cd119.A*cycle_time;
CbndCd119 = ld*Cd119.halflife*Cd119.amount*NA/Cd119.A*cycle_time;
In119.amount += 1.00*PbndCd119/NA*Cd119.A;
PbndCd120 = ld*Cd120.halflife*Cd120.amount*NA/Cd120.A*cycle_time;
CbndCd120 = ld*Cd120.halflife*Cd120.amount*NA/Cd120.A*cycle_time;
In120.amount += 1.00*PbndCd120/NA*Cd120.A;
PbndCd121 = ld*Cd121.halflife*Cd121.amount*NA/Cd121.A*cycle_time;
CbndCd121 = ld*Cd121.halflife*Cd121.amount*NA/Cd121.A*cycle_time;
In121.amount += 1.00*PbndCd121/NA*Cd121.A;
PbndCd122 = ld*Cd122.halflife*Cd122.amount*NA/Cd122.A*cycle_time;
CbndCd122 = ld*Cd122.halflife*Cd122.amount*NA/Cd122.A*cycle_time;
In122.amount += 1.00*PbndCd122/NA*Cd122.A;
PbndCd123 = ld*Cd123.halflife*Cd123.amount*NA/Cd123.A*cycle_time;
CbndCd123 = ld*Cd123.halflife*Cd123.amount*NA/Cd123.A*cycle_time;
In123.amount += 1.00*PbndCd123/NA*Cd123.A;
PbndCd124 = ld*Cd124.halflife*Cd124.amount*NA/Cd124.A*cycle_time;
```

```
CbndCd124 = ld*Cd124.halflife*Cd124.amount*NA/Cd124.A*cycle_time;
In124.amount += 1.00*PbndCd124/NA*Cd124.A;
PbndCd125 = ld*Cd125.halflife*Cd125.amount*NA/Cd125.A*cycle_time;
CbndCd125 = ld*Cd125.halflife*Cd125.amount*NA/Cd125.A*cycle_time;
In125.amount += 1.00*PbndCd125/NA*Cd125.A;
PbndCd126 = ld*Cd126.halflife*Cd126.amount*NA/Cd126.A*cycle_time;
CbndCd126 = ld*Cd126.halflife*Cd126.amount*NA/Cd126.A*cycle_time;
In126.amount += 1.00*PbndCd126/NA*Cd126.A;
PbndCd127 = ld*Cd127.halflife*Cd127.amount*NA/Cd127.A*cycle_time;
CbndCd127 = ld*Cd127.halflife*Cd127.amount*NA/Cd127.A*cycle_time;
In127.amount += 1.00*PbndCd127/NA*Cd127.A;
PbndCd128 = ld*Cd128.halflife*Cd128.amount*NA/Cd128.A*cycle_time;
CbndCd128 = ld*Cd128.halflife*Cd128.amount*NA/Cd128.A*cycle_time;
In128.amount += 1.00*PbndCd128/NA*Cd128.A;
PbndCd129 = ld*Cd129.halflife*Cd129.amount*NA/Cd129.A*cycle_time;
CbndCd129 = ld*Cd129.halflife*Cd129.amount*NA/Cd129.A*cycle_time;
In129.amount += 1.00*PbndCd129/NA*Cd129.A;
PbndCd130 = ld*Cd130.halflife*Cd130.amount*NA/Cd130.A*cycle_time;
CbndCd130 = ld*Cd130.halflife*Cd130.amount*NA/Cd130.A*cycle_time;
In130.amount += 1.00*PbndCd130/NA*Cd130.A;
PbndAg112 = ld*Ag112.halflife*Ag112.amount*NA/Ag112.A*cycle_time;
CbndAg112 = ld*Ag112.halflife*Ag112.amount*NA/Ag112.A*cycle_time;
Cd112.amount += 1.00*PbndAg112/NA*Ag112.A;
PbndAg113 = ld*Ag113.halflife*Ag113.amount*NA/Ag113.A*cycle_time;
CbndAg113 = ld*Ag113.halflife*Ag113.amount*NA/Ag113.A*cycle_time;
Cd113.amount += 1.00*PbndAg113/NA*Ag113.A;
PbndAg114 = ld*Ag114.halflife*Ag114.amount*NA/Ag114.A*cycle_time;
CbndAg114 = ld*Ag114.halflife*Ag114.amount*NA/Ag114.A*cycle_time;
Cd114.amount += 1.00*PbndAg114/NA*Ag114.A;
PbndAg115 = ld*Ag115.halflife*Ag115.amount*NA/Ag115.A*cycle_time;
CbndAg115 = ld*Ag115.halflife*Ag115.amount*NA/Ag115.A*cycle_time;
Cd115.amount += 1.00*PbndAg115/NA*Ag115.A;
```

```
PbndAg116 = ld*Ag116.halflife*Ag116.amount*NA/Ag116.A*cycle_time;
CbndAg116 = ld*Ag116.halflife*Ag116.amount*NA/Ag116.A*cycle_time;
Cd116.amount += 1.00*PbndAg116/NA*Ag116.A;
PbndAg117 = ld*Ag117.halflife*Ag117.amount*NA/Ag117.A*cycle_time;
CbndAg117 = ld*Ag117.halflife*Ag117.amount*NA/Ag117.A*cycle_time;
Cd117.amount += 1.00*PbndAg117/NA*Ag117.A;
PbndAg118 = ld*Ag118.halflife*Ag118.amount*NA/Ag118.A*cycle_time;
CbndAg118 = ld*Ag118.halflife*Ag118.amount*NA/Ag118.A*cycle_time;
Cd118.amount += 1.00*PbndAg118/NA*Ag118.A;
PbndAg119 = ld*Ag119.halflife*Ag119.amount*NA/Ag119.A*cycle_time;
CbndAg119 = ld*Ag119.halflife*Ag119.amount*NA/Ag119.A*cycle_time;
Cd119.amount += 1.00*PbndAg119/NA*Ag119.A;
PbndAg120 = ld*Ag120.halflife*Ag120.amount*NA/Ag120.A*cycle_time;
CbndAg120 = ld*Ag120.halflife*Ag120.amount*NA/Ag120.A*cycle_time;
Cd120.amount += 1.00*PbndAg120/NA*Ag120.A;
PbndAg121 = ld*Ag121.halflife*Ag121.amount*NA/Ag121.A*cycle_time;
CbndAg121 = ld*Ag121.halflife*Ag121.amount*NA/Ag121.A*cycle_time;
Cd121.amount += 1.00*PbndAg121/NA*Ag121.A;
PbndAg122 = ld*Ag122.halflife*Ag122.amount*NA/Ag122.A*cycle_time;
CbndAg122 = ld*Ag122.halflife*Ag122.amount*NA/Ag122.A*cycle_time;
Cd122.amount += 1.00*PbndAg122/NA*Ag122.A;
PbndAg123 = ld*Ag123.halflife*Ag123.amount*NA/Ag123.A*cycle_time;
CbndAg123 = ld*Ag123.halflife*Ag123.amount*NA/Ag123.A*cycle_time;
Cd123.amount += 1.00*PbndAg123/NA*Ag123.A;
PbndAg124 = ld*Ag124.halflife*Ag124.amount*NA/Ag124.A*cycle_time;
CbndAg124 = ld*Ag124.halflife*Ag124.amount*NA/Ag124.A*cycle_time;
Cd124.amount += 1.00*PbndAg124/NA*Ag124.A;
PbndPd111 = ld*Pd111.halflife*Pd111.amount*NA/Pd111.A*cycle_time;
CbndPd111 = ld*Pd111.halflife*Pd111.amount*NA/Pd111.A*cycle_time;
Ag111.amount += 1.00*PbndPd111/NA*Pd111.A;
PbndPd112 = ld*Pd112.halflife*Pd112.amount*NA/Pd112.A*cycle_time;
CbndPd112 = ld*Pd112.halflife*Pd112.amount*NA/Pd112.A*cycle_time;
```

```
Ag112.amount += 1.00*PbndPd112/NA*Pd112.A;
PbndPd113 = ld*Pd113.halflife*Pd113.amount*NA/Pd113.A*cycle_time;
CbndPd113 = ld*Pd113.halflife*Pd113.amount*NA/Pd113.A*cycle_time;
Ag113.amount += 1.00*PbndPd113/NA*Pd113.A;
PbndPd114 = ld*Pd114.halflife*Pd114.amount*NA/Pd114.A*cycle_time;
CbndPd114 = ld*Pd114.halflife*Pd114.amount*NA/Pd114.A*cycle_time;
Ag114.amount += 1.00*PbndPd114/NA*Pd114.A;
PbndPd115 = ld*Pd115.halflife*Pd115.amount*NA/Pd115.A*cycle_time;
CbndPd115 = ld*Pd115.halflife*Pd115.amount*NA/Pd115.A*cycle_time;
Ag115.amount += 1.00*PbndPd115/NA*Pd115.A;
PbndPd116 = ld*Pd116.halflife*Pd116.amount*NA/Pd116.A*cycle_time;
CbndPd116 = ld*Pd116.halflife*Pd116.amount*NA/Pd116.A*cycle_time;
Ag116.amount += 1.00*PbndPd116/NA*Pd116.A;
PbndPd117 = ld*Pd117.halflife*Pd117.amount*NA/Pd117.A*cycle_time;
CbndPd117 = ld*Pd117.halflife*Pd117.amount*NA/Pd117.A*cycle_time;
Ag117.amount += 1.00*PbndPd117/NA*Pd117.A;
PbndPd118 = ld*Pd118.halflife*Pd118.amount*NA/Pd118.A*cycle_time;
CbndPd118 = ld*Pd118.halflife*Pd118.amount*NA/Pd118.A*cycle_time;
Ag118.amount += 1.00*PbndPd118/NA*Pd118.A;
PbndRh107 = ld*Rh107.halflife*Rh107.amount*NA/Rh107.A*cycle_time;
CbndRh107 = ld*Rh107.halflife*Rh107.amount*NA/Rh107.A*cycle_time;
Pd107.amount += 1.00*PbndRh107/NA*Rh107.A;
PbndRh108 = ld*Rh108.halflife*Rh108.amount*NA/Rh108.A*cycle_time;
CbndRh108 = ld*Rh108.halflife*Rh108.amount*NA/Rh108.A*cycle_time;
Pd108.amount += 1.00*PbndRh108/NA*Rh108.A;
PbndRh109 = ld*Rh109.halflife*Rh109.amount*NA/Rh109.A*cycle_time;
CbndRh109 = ld*Rh109.halflife*Rh109.amount*NA/Rh109.A*cycle_time;
Pd109.amount += 1.00*PbndRh109/NA*Rh109.A;
PbndRh110 = ld*Rh110.halflife*Rh110.amount*NA/Rh110.A*cycle_time;
CbndRh110 = ld*Rh110.halflife*Rh110.amount*NA/Rh110.A*cycle_time;
Pd110.amount += 1.00*PbndRh110/NA*Rh110.A;
PbndRh111 = ld*Rh111.halflife*Rh111.amount*NA/Rh111.A*cycle_time;
```

```
CbndRh111 = ld*Rh111.halflife*Rh111.amount*NA/Rh111.A*cycle_time;
Pd111.amount += 1.00*PbndRh111/NA*Rh111.A;
PbndRh112 = ld*Rh112.halflife*Rh112.amount*NA/Rh112.A*cycle_time;
CbndRh112 = ld*Rh112.halflife*Rh112.amount*NA/Rh112.A*cycle_time;
Pd112.amount += 1.00*PbndRh112/NA*Rh112.A;
PbndRh113 = ld*Rh113.halflife*Rh113.amount*NA/Rh113.A*cycle_time;
CbndRh113 = ld*Rh113.halflife*Rh113.amount*NA/Rh113.A*cycle_time;
Pd113.amount += 1.00*PbndRh113/NA*Rh113.A;
PbndRh114 = ld*Rh114.halflife*Rh114.amount*NA/Rh114.A*cycle_time;
CbndRh114 = ld*Rh114.halflife*Rh114.amount*NA/Rh114.A*cycle_time;
Pd114.amount += 1.00*PbndRh114/NA*Rh114.A;
PbndRh115 = ld*Rh115.halflife*Rh115.amount*NA/Rh115.A*cycle_time;
CbndRh115 = ld*Rh115.halflife*Rh115.amount*NA/Rh115.A*cycle_time;
Pd115.amount += 1.00*PbndRh115/NA*Rh115.A;
PbndRh116 = ld*Rh116.halflife*Rh116.amount*NA/Rh116.A*cycle_time;
CbndRh116 = ld*Rh116.halflife*Rh116.amount*NA/Rh116.A*cycle_time;
Pd116.amount += 1.00*PbndRh116/NA*Rh116.A;
PbndRu105 = ld*Ru105.halflife*Ru105.amount*NA/Ru105.A*cycle_time;
CbndRu105 = ld*Ru105.halflife*Ru105.amount*NA/Ru105.A*cycle_time;
Rh105.amount += 1.00*PbndRu105/NA*Ru105.A;
PbndRu106 = ld*Ru106.halflife*Ru106.amount*NA/Ru106.A*cycle_time;
CbndRu106 = ld*Ru106.halflife*Ru106.amount*NA/Ru106.A*cycle_time;
Rh106.amount += 1.00*PbndRu106/NA*Ru106.A;
PbndRu107 = ld*Ru107.halflife*Ru107.amount*NA/Ru107.A*cycle_time;
CbndRu107 = ld*Ru107.halflife*Ru107.amount*NA/Ru107.A*cycle_time;
Rh107.amount += 1.00*PbndRu107/NA*Ru107.A;
PbndRu108 = ld*Ru108.halflife*Ru108.amount*NA/Ru108.A*cycle_time;
CbndRu108 = ld*Ru108.halflife*Ru108.amount*NA/Ru108.A*cycle_time;
Rh108.amount += 1.00*PbndRu108/NA*Ru108.A;
PbndRu109 = ld*Ru109.halflife*Ru109.amount*NA/Ru109.A*cycle_time;
CbndRu109 = ld*Ru109.halflife*Ru109.amount*NA/Ru109.A*cycle_time;
Rh109.amount += 1.00*PbndRu109/NA*Ru109.A;
```

```
PbndRu110 = ld*Ru110.halflife*Ru110.amount*NA/Ru110.A*cycle_time;
CbndRu110 = ld*Ru110.halflife*Ru110.amount*NA/Ru110.A*cycle_time;
Rh110.amount += 1.00*PbndRu110/NA*Ru110.A;
PbndRu111 = ld*Ru111.halflife*Ru111.amount*NA/Ru111.A*cycle_time;
CbndRu111 = ld*Ru111.halflife*Ru111.amount*NA/Ru111.A*cycle_time;
Rh111.amount += 1.00*PbndRu111/NA*Ru111.A;
PbndRu112 = ld*Ru112.halflife*Ru112.amount*NA/Ru112.A*cycle_time;
CbndRu112 = ld*Ru112.halflife*Ru112.amount*NA/Ru112.A*cycle_time;
Rh112.amount += 1.00*PbndRu112/NA*Ru112.A;
PbndRu113 = ld*Ru113.halflife*Ru113.amount*NA/Ru113.A*cycle_time;
CbndRu113 = ld*Ru113.halflife*Ru113.amount*NA/Ru113.A*cycle_time;
Rh113.amount += 1.00*PbndRu113/NA*Ru113.A;
PbndTc102 = ld*Tc102.halflife*Tc102.amount*NA/Tc102.A*cycle_time;
CbndTc102 = ld*Tc102.halflife*Tc102.amount*NA/Tc102.A*cycle_time;
Ru102.amount += 1.00*PbndTc102/NA*Tc102.A;
PbndTc103 = ld*Tc103.halflife*Tc103.amount*NA/Tc103.A*cycle_time;
CbndTc103 = ld*Tc103.halflife*Tc103.amount*NA/Tc103.A*cycle_time;
Ru103.amount += 1.00*PbndTc103/NA*Tc103.A;
PbndTc104 = ld*Tc104.halflife*Tc104.amount*NA/Tc104.A*cycle_time;
CbndTc104 = ld*Tc104.halflife*Tc104.amount*NA/Tc104.A*cycle_time;
Ru104.amount += 1.00*PbndTc104/NA*Tc104.A;
PbndTc105 = ld*Tc105.halflife*Tc105.amount*NA/Tc105.A*cycle_time;
CbndTc105 = ld*Tc105.halflife*Tc105.amount*NA/Tc105.A*cycle_time;
Ru105.amount += 1.00*PbndTc105/NA*Tc105.A;
PbndTc106 = ld*Tc106.halflife*Tc106.amount*NA/Tc106.A*cycle_time;
CbndTc106 = ld*Tc106.halflife*Tc106.amount*NA/Tc106.A*cycle_time;
Ru106.amount += 1.00*PbndTc106/NA*Tc106.A;
PbndTc107 = ld*Tc107.halflife*Tc107.amount*NA/Tc107.A*cycle_time;
CbndTc107 = ld*Tc107.halflife*Tc107.amount*NA/Tc107.A*cycle_time;
Ru107.amount += 1.00*PbndTc107/NA*Tc107.A;
PbndTc108 = ld*Tc108.halflife*Tc108.amount*NA/Tc108.A*cycle_time;
CbndTc108 = ld*Tc108.halflife*Tc108.amount*NA/Tc108.A*cycle_time;
```

```
Ru108.amount += 1.00*PbndTc108/NA*Tc108.A;
PbndTc109 = ld*Tc109.halflife*Tc109.amount*NA/Tc109.A*cycle_time;
CbndTc109 = ld*Tc109.halflife*Tc109.amount*NA/Tc109.A*cycle_time;
Ru109.amount += 1.00*PbndTc109/NA*Tc109.A;
PbndTc110 = ld*Tc110.halflife*Tc110.amount*NA/Tc110.A*cycle_time;
CbndTc110 = ld*Tc110.halflife*Tc110.amount*NA/Tc110.A*cycle_time;
Ru110.amount += 1.00*PbndTc110/NA*Tc110.A;
PbndTc111 = ld*Tc111.halflife*Tc111.amount*NA/Tc111.A*cycle_time;
CbndTc111 = ld*Tc111.halflife*Tc111.amount*NA/Tc111.A*cycle_time;
Ru111.amount += 1.00*PbndTc111/NA*Tc111.A;
FgMo100 = Mo100.amount*Mo100.sigma_g/Sigma_a_fuel;
nrcMo100 = neutron.amount*k/eta*FgMo100;
PrcMo100 = nrcMo100*Mo100.A/NA;
CrcMo100 = nrcMo100*Mo100.A/NA;
Mo100.amount += -1.00*CrcMo100+1.00*PbndNb100/NA*Nb100.A;
Mo101.amount += 1.00*PrcMo100+1.00*PbndNb101/NA*Nb101.A;
PbndMo101 = ld*Mo101.halflife*Mo101.amount*NA/Mo101.A*cycle_time;
CbndMo101 = ld*Mo101.halflife*Mo101.amount*NA/Mo101.A*cycle_time;
Tc101.amount += 1.00*PbndMo101/NA*Mo101.A;
PbndMo102 = ld*Mo102.halflife*Mo102.amount*NA/Mo102.A*cycle_time;
CbndMo102 = ld*Mo102.halflife*Mo102.amount*NA/Mo102.A*cycle_time;
Tc102.amount += 1.00*PbndMo102/NA*Mo102.A;
PbndMo103 = ld*Mo103.halflife*Mo103.amount*NA/Mo103.A*cycle_time;
CbndMo103 = ld*Mo103.halflife*Mo103.amount*NA/Mo103.A*cycle_time;
Tc103.amount += 1.00*PbndMo103/NA*Mo103.A;
PbndMo104 = ld*Mo104.halflife*Mo104.amount*NA/Mo104.A*cycle_time;
CbndMo104 = ld*Mo104.halflife*Mo104.amount*NA/Mo104.A*cycle_time;
Tc104.amount += 1.00*PbndMo104/NA*Mo104.A;
PbndMo105 = ld*Mo105.halflife*Mo105.amount*NA/Mo105.A*cycle_time;
CbndMo105 = ld*Mo105.halflife*Mo105.amount*NA/Mo105.A*cycle_time;
Tc105.amount += 1.00*PbndMo105/NA*Mo105.A;
PbndMo106 = ld*Mo106.halflife*Mo106.amount*NA/Mo106.A*cycle_time;
```

```
CbndMo106 = ld*Mo106.halflife*Mo106.amount*NA/Mo106.A*cycle_time;
Tc106.amount += 1.00*PbndMo106/NA*Mo106.A;
PbndMo107 = ld*Mo107.halflife*Mo107.amount*NA/Mo107.A*cycle_time;
CbndMo107 = ld*Mo107.halflife*Mo107.amount*NA/Mo107.A*cycle_time;
Tc107.amount += 1.00*PbndMo107/NA*Mo107.A;
PbndNb97 = ld*Nb97.halflife*Nb97.amount*NA/Nb97.A*cycle_time;
CbndNb97 = ld*Nb97.halflife*Nb97.amount*NA/Nb97.A*cycle_time;
Mo97.amount += 1.00*PbndNb97/NA*Nb97.A;
PbndNb98 = ld*Nb98.halflife*Nb98.amount*NA/Nb98.A*cycle_time;
CbndNb98 = ld*Nb98.halflife*Nb98.amount*NA/Nb98.A*cycle_time;
Mo98.amount += 1.00*PbndNb98/NA*Nb98.A;
PbndNb99 = ld*Nb99.halflife*Nb99.amount*NA/Nb99.A*cycle_time;
CbndNb99 = ld*Nb99.halflife*Nb99.amount*NA/Nb99.A*cycle_time;
Mo99.amount += 1.00*PbndNb99/NA*Nb99.A;
PbndNb100 = ld*Nb100.halflife*Nb100.amount*NA/Nb100.A*cycle_time;
CbndNb100 = ld*Nb100.halflife*Nb100.amount*NA/Nb100.A*cycle_time;
PbndNb101 = ld*Nb101.halflife*Nb101.amount*NA/Nb101.A*cycle_time;
CbndNb101 = ld*Nb101.halflife*Nb101.amount*NA/Nb101.A*cycle_time;
PbndNb102 = ld*Nb102.halflife*Nb102.amount*NA/Nb102.A*cycle_time;
CbndNb102 = ld*Nb102.halflife*Nb102.amount*NA/Nb102.A*cycle_time;
Mo102.amount += 1.00*PbndNb102/NA*Nb102.A;
PbndNb103 = ld*Nb103.halflife*Nb103.amount*NA/Nb103.A*cycle_time;
CbndNb103 = ld*Nb103.halflife*Nb103.amount*NA/Nb103.A*cycle_time;
Mo103.amount += 1.00*PbndNb103/NA*Nb103.A;
PbndNb104 = ld*Nb104.halflife*Nb104.amount*NA/Nb104.A*cycle_time;
CbndNb104 = ld*Nb104.halflife*Nb104.amount*NA/Nb104.A*cycle_time;
Mo104.amount += 1.00*PbndNb104/NA*Nb104.A;
PbndNb105 = ld*Nb105.halflife*Nb105.amount*NA/Nb105.A*cycle_time;
CbndNb105 = ld*Nb105.halflife*Nb105.amount*NA/Nb105.A*cycle_time;
Mo105.amount += 1.00*PbndNb105/NA*Nb105.A;
PbndNb106 = ld*Nb106.halflife*Nb106.amount*NA/Nb106.A*cycle_time;
CbndNb106 = ld*Nb106.halflife*Nb106.amount*NA/Nb106.A*cycle_time;
```

```
Mo106.amount += 1.00*PbndNb106/NA*Nb106.A;
PbndZr95 = ld*Zr95.halflife*Zr95.amount*NA/Zr95.A*cycle_time;
CbndZr95 = ld*Zr95.halflife*Zr95.amount*NA/Zr95.A*cycle_time;
Nb95.amount += 1.00*PbndZr95/NA*Zr95.A;
PbndZr97 = ld*Zr97.halflife*Zr97.amount*NA/Zr97.A*cycle_time;
CbndZr97 = ld*Zr97.halflife*Zr97.amount*NA/Zr97.A*cycle_time;
Nb97.amount += 1.00*PbndZr97/NA*Zr97.A;
PbndZr98 = ld*Zr98.halflife*Zr98.amount*NA/Zr98.A*cycle_time;
CbndZr98 = ld*Zr98.halflife*Zr98.amount*NA/Zr98.A*cycle_time;
Nb98.amount += 1.00*PbndZr98/NA*Zr98.A;
PbndZr99 = ld*Zr99.halflife*Zr99.amount*NA/Zr99.A*cycle_time;
CbndZr99 = ld*Zr99.halflife*Zr99.amount*NA/Zr99.A*cycle_time;
Nb99.amount += 1.00*PbndZr99/NA*Zr99.A;
PbndZr100 = ld*Zr100.halflife*Zr100.amount*NA/Zr100.A*cycle_time;
CbndZr100 = ld*Zr100.halflife*Zr100.amount*NA/Zr100.A*cycle_time;
Nb100.amount += 1.00*PbndZr100/NA*Zr100.A;
PbndZr101 = ld*Zr101.halflife*Zr101.amount*NA/Zr101.A*cycle_time;
CbndZr101 = ld*Zr101.halflife*Zr101.amount*NA/Zr101.A*cycle_time;
Nb101.amount += 1.00*PbndZr101/NA*Zr101.A;
PbndZr102 = ld*Zr102.halflife*Zr102.amount*NA/Zr102.A*cycle_time;
CbndZr102 = ld*Zr102.halflife*Zr102.amount*NA/Zr102.A*cycle_time;
Nb102.amount += 1.00*PbndZr102/NA*Zr102.A;
PbndZr103 = ld*Zr103.halflife*Zr103.amount*NA/Zr103.A*cycle_time;
CbndZr103 = ld*Zr103.halflife*Zr103.amount*NA/Zr103.A*cycle_time;
Nb103.amount += 1.00*PbndZr103/NA*Zr103.A;
PbndZr104 = ld*Zr104.halflife*Zr104.amount*NA/Zr104.A*cycle_time;
CbndZr104 = ld*Zr104.halflife*Zr104.amount*NA/Zr104.A*cycle_time;
Nb104.amount += 1.00*PbndZr104/NA*Zr104.A;
PbndY92 = ld*Y92.halflife*Y92.amount*NA/Y92.A*cycle_time;
CbndY92 = ld*Y92.halflife*Y92.amount*NA/Y92.A*cycle_time;
Zr92.amount += 1.00*PbndY92/NA*Y92.A;
PbndY93 = ld*Y93.halflife*Y93.amount*NA/Y93.A*cycle_time;
```

```
CbndY93 = ld*Y93.halflife*Y93.amount*NA/Y93.A*cycle_time;
Zr93.amount += 1.00*PbndY93/NA*Y93.A;
PbndY94 = ld*Y94.halflife*Y94.amount*NA/Y94.A*cycle_time;
CbndY94 = ld*Y94.halflife*Y94.amount*NA/Y94.A*cycle_time;
Zr94.amount += 1.00*PbndY94/NA*Y94.A;
PbndY95 = ld*Y95.halflife*Y95.amount*NA/Y95.A*cycle_time;
CbndY95 = ld*Y95.halflife*Y95.amount*NA/Y95.A*cycle_time;
Zr95.amount += 1.00*PbndY95/NA*Y95.A;
PbndY96 = ld*Y96.halflife*Y96.amount*NA/Y96.A*cycle_time;
CbndY96 = ld*Y96.halflife*Y96.amount*NA/Y96.A*cycle_time;
Zr96.amount += 1.00*PbndY96/NA*Y96.A;
PbndY97 = ld*Y97.halflife*Y97.amount*NA/Y97.A*cycle_time;
CbndY97 = ld*Y97.halflife*Y97.amount*NA/Y97.A*cycle_time;
Zr97.amount += 1.00*PbndY97/NA*Y97.A;
PbndY98 = ld*Y98.halflife*Y98.amount*NA/Y98.A*cycle_time;
CbndY98 = ld*Y98.halflife*Y98.amount*NA/Y98.A*cycle_time;
Zr98.amount += 1.00*PbndY98/NA*Y98.A;
PbndY99 = ld*Y99.halflife*Y99.amount*NA/Y99.A*cycle_time;
CbndY99 = ld*Y99.halflife*Y99.amount*NA/Y99.A*cycle_time;
Zr99.amount += 1.00*PbndY99/NA*Y99.A;
PbndY100 = ld*Y100.halflife*Y100.amount*NA/Y100.A*cycle_time;
CbndY100 = ld*Y100.halflife*Y100.amount*NA/Y100.A*cycle_time;
Zr100.amount += 1.00*PbndY100/NA*Y100.A;
PbndY101 = ld*Y101.halflife*Y101.amount*NA/Y101.A*cycle_time;
CbndY101 = ld*Y101.halflife*Y101.amount*NA/Y101.A*cycle_time;
Zr101.amount += 1.00*PbndY101/NA*Y101.A;
PbndY102 = ld*Y102.halflife*Y102.amount*NA/Y102.A*cycle_time;
CbndY102 = ld*Y102.halflife*Y102.amount*NA/Y102.A*cycle_time;
Zr102.amount += 1.00*PbndY102/NA*Y102.A;
PbndSr89 = ld*Sr89.halflife*Sr89.amount*NA/Sr89.A*cycle_time;
CbndSr89 = ld*Sr89.halflife*Sr89.amount*NA/Sr89.A*cycle_time;
Y89.amount += 1.00*PbndSr89/NA*Sr89.A;
```

```
PbndSr90 = ld*Sr90.halflife*Sr90.amount*NA/Sr90.A*cycle_time;
CbndSr90 = ld*Sr90.halflife*Sr90.amount*NA/Sr90.A*cycle_time;
Y90.amount += 1.00*PbndSr90/NA*Sr90.A;
PbndSr91 = ld*Sr91.halflife*Sr91.amount*NA/Sr91.A*cycle_time;
CbndSr91 = ld*Sr91.halflife*Sr91.amount*NA/Sr91.A*cycle_time;
Y91.amount += 1.00*PbndSr91/NA*Sr91.A;
PbndSr92 = ld*Sr92.halflife*Sr92.amount*NA/Sr92.A*cycle_time;
CbndSr92 = ld*Sr92.halflife*Sr92.amount*NA/Sr92.A*cycle_time;
Y92.amount += 1.00*PbndSr92/NA*Sr92.A;
PbndSr93 = ld*Sr93.halflife*Sr93.amount*NA/Sr93.A*cycle_time;
CbndSr93 = ld*Sr93.halflife*Sr93.amount*NA/Sr93.A*cycle_time;
Y93.amount += 1.00*PbndSr93/NA*Sr93.A;
PbndSr94 = ld*Sr94.halflife*Sr94.amount*NA/Sr94.A*cycle_time;
CbndSr94 = ld*Sr94.halflife*Sr94.amount*NA/Sr94.A*cycle_time;
Y94.amount += 1.00*PbndSr94/NA*Sr94.A;
PbndSr95 = ld*Sr95.halflife*Sr95.amount*NA/Sr95.A*cycle_time;
CbndSr95 = ld*Sr95.halflife*Sr95.amount*NA/Sr95.A*cycle_time;
Y95.amount += 1.00*PbndSr95/NA*Sr95.A;
PbndSr96 = ld*Sr96.halflife*Sr96.amount*NA/Sr96.A*cycle_time;
CbndSr96 = ld*Sr96.halflife*Sr96.amount*NA/Sr96.A*cycle_time;
Y96.amount += 1.00*PbndSr96/NA*Sr96.A;
PbndSr97 = ld*Sr97.halflife*Sr97.amount*NA/Sr97.A*cycle_time;
CbndSr97 = ld*Sr97.halflife*Sr97.amount*NA/Sr97.A*cycle_time;
Y97.amount += 1.00*PbndSr97/NA*Sr97.A;
PbndSr98 = ld*Sr98.halflife*Sr98.amount*NA/Sr98.A*cycle_time;
CbndSr98 = ld*Sr98.halflife*Sr98.amount*NA/Sr98.A*cycle_time;
Y98.amount += 1.00*PbndSr98/NA*Sr98.A;
PbndSr99 = ld*Sr99.halflife*Sr99.amount*NA/Sr99.A*cycle_time;
CbndSr99 = ld*Sr99.halflife*Sr99.amount*NA/Sr99.A*cycle_time;
Y99.amount += 1.00*PbndSr99/NA*Sr99.A;
PbndSr100 = ld*Sr100.halflife*Sr100.amount*NA/Sr100.A*cycle_time;
CbndSr100 = ld*Sr100.halflife*Sr100.amount*NA/Sr100.A*cycle_time;
```

```
Y100.amount += 1.00*PbndSr100/NA*Sr100.A;
PbndSr101 = ld*Sr101.halflife*Sr101.amount*NA/Sr101.A*cycle_time;
CbndSr101 = ld*Sr101.halflife*Sr101.amount*NA/Sr101.A*cycle_time;
Y101.amount += 1.00*PbndSr101/NA*Sr101.A;
PbndSr102 = ld*Sr102.halflife*Sr102.amount*NA/Sr102.A*cycle_time;
CbndSr102 = ld*Sr102.halflife*Sr102.amount*NA/Sr102.A*cycle_time;
Y102.amount += 1.00*PbndSr102/NA*Sr102.A;
PbndRb88 = ld*Rb88.halflife*Rb88.amount*NA/Rb88.A*cycle_time;
CbndRb88 = ld*Rb88.halflife*Rb88.amount*NA/Rb88.A*cycle_time;
Sr88.amount += 1.00*PbndRb88/NA*Rb88.A;
PbndRb89 = ld*Rb89.halflife*Rb89.amount*NA/Rb89.A*cycle_time;
CbndRb89 = ld*Rb89.halflife*Rb89.amount*NA/Rb89.A*cycle_time;
Sr89.amount += 1.00*PbndRb89/NA*Rb89.A;
PbndRb90 = ld*Rb90.halflife*Rb90.amount*NA/Rb90.A*cycle_time;
CbndRb90 = ld*Rb90.halflife*Rb90.amount*NA/Rb90.A*cycle_time;
Sr90.amount += 1.00*PbndRb90/NA*Rb90.A;
PbndRb91 = ld*Rb91.halflife*Rb91.amount*NA/Rb91.A*cycle_time;
CbndRb91 = ld*Rb91.halflife*Rb91.amount*NA/Rb91.A*cycle_time;
Sr91.amount += 1.00*PbndRb91/NA*Rb91.A;
PbndRb92 = ld*Rb92.halflife*Rb92.amount*NA/Rb92.A*cycle_time;
CbndRb92 = ld*Rb92.halflife*Rb92.amount*NA/Rb92.A*cycle_time;
Sr92.amount += 1.00*PbndRb92/NA*Rb92.A;
PbndRb93 = ld*Rb93.halflife*Rb93.amount*NA/Rb93.A*cycle_time;
CbndRb93 = ld*Rb93.halflife*Rb93.amount*NA/Rb93.A*cycle_time;
Sr93.amount += 1.00*PbndRb93/NA*Rb93.A;
PbndRb94 = ld*Rb94.halflife*Rb94.amount*NA/Rb94.A*cycle_time;
CbndRb94 = ld*Rb94.halflife*Rb94.amount*NA/Rb94.A*cycle_time;
Sr94.amount += 1.00*PbndRb94/NA*Rb94.A;
PbndRb95 = ld*Rb95.halflife*Rb95.amount*NA/Rb95.A*cycle_time;
CbndRb95 = ld*Rb95.halflife*Rb95.amount*NA/Rb95.A*cycle_time;
Sr95.amount += 1.00*PbndRb95/NA*Rb95.A;
PbndRb96 = ld*Rb96.halflife*Rb96.amount*NA/Rb96.A*cycle_time;
```

```
CbndRb96 = ld*Rb96.halflife*Rb96.amount*NA/Rb96.A*cycle_time;
Sr96.amount += 1.00*PbndRb96/NA*Rb96.A;
PbndRb97 = ld*Rb97.halflife*Rb97.amount*NA/Rb97.A*cycle_time;
CbndRb97 = ld*Rb97.halflife*Rb97.amount*NA/Rb97.A*cycle_time;
Sr97.amount += 1.00*PbndRb97/NA*Rb97.A;
PbndRb98 = ld*Rb98.halflife*Rb98.amount*NA/Rb98.A*cycle_time;
CbndRb98 = ld*Rb98.halflife*Rb98.amount*NA/Rb98.A*cycle_time;
Sr98.amount += 1.00*PbndRb98/NA*Rb98.A;
PbndRb99 = ld*Rb99.halflife*Rb99.amount*NA/Rb99.A*cycle_time;
CbndRb99 = ld*Rb99.halflife*Rb99.amount*NA/Rb99.A*cycle_time;
Sr99.amount += 1.00*PbndRb99/NA*Rb99.A;
PbndRb100 = ld*Rb100.halflife*Rb100.amount*NA/Rb100.A*cycle_time;
CbndRb100 = ld*Rb100.halflife*Rb100.amount*NA/Rb100.A*cycle_time;
Sr100.amount += 1.00*PbndRb100/NA*Rb100.A;
PbndRb102 = ld*Rb102.halflife*Rb102.amount*NA/Rb102.A*cycle_time;
CbndRb102 = ld*Rb102.halflife*Rb102.amount*NA/Rb102.A*cycle_time;
Sr102.amount += 1.00*PbndRb102/NA*Rb102.A;
FgKr85 = Kr85.amount*Kr85.sigma_g/Sigma_a_fuel;
nrcKr85 = neutron.amount*k/eta*FgKr85;
PrcKr85 = nrcKr85*Kr85.A/NA;
CrcKr85 = nrcKr85*Kr85.A/NA;
Kr85.amount += -1.00*CrcKr85+1.00*PbndBr85/NA*Br85.A;
Kr86.amount += 1.00*PrcKr85+1.00*PbndBr86/NA*Br86.A;
PbndKr85 = ld*Kr85.halflife*Kr85.amount*NA/Kr85.A*cycle_time;
CbndKr85 = ld*Kr85.halflife*Kr85.amount*NA/Kr85.A*cycle_time;
Rb85.amount += 1.00*PbndKr85/NA*Kr85.A;
PbndKr87 = ld*Kr87.halflife*Kr87.amount*NA/Kr87.A*cycle_time;
CbndKr87 = ld*Kr87.halflife*Kr87.amount*NA/Kr87.A*cycle_time;
Rb87.amount += 1.00*PbndKr87/NA*Kr87.A;
PbndKr88 = ld*Kr88.halflife*Kr88.amount*NA/Kr88.A*cycle_time;
CbndKr88 = ld*Kr88.halflife*Kr88.amount*NA/Kr88.A*cycle_time;
Rb88.amount += 1.00*PbndKr88/NA*Kr88.A;
```

```
PbndKr89 = ld*Kr89.halflife*Kr89.amount*NA/Kr89.A*cycle_time;
CbndKr89 = ld*Kr89.halflife*Kr89.amount*NA/Kr89.A*cycle_time;
Rb89.amount += 1.00*PbndKr89/NA*Kr89.A;
PbndKr90 = ld*Kr90.halflife*Kr90.amount*NA/Kr90.A*cycle_time;
CbndKr90 = ld*Kr90.halflife*Kr90.amount*NA/Kr90.A*cycle_time;
Rb90.amount += 1.00*PbndKr90/NA*Kr90.A;
PbndKr91 = ld*Kr91.halflife*Kr91.amount*NA/Kr91.A*cycle_time;
CbndKr91 = ld*Kr91.halflife*Kr91.amount*NA/Kr91.A*cycle_time;
Rb91.amount += 1.00*PbndKr91/NA*Kr91.A;
PbndKr92 = ld*Kr92.halflife*Kr92.amount*NA/Kr92.A*cycle_time;
CbndKr92 = ld*Kr92.halflife*Kr92.amount*NA/Kr92.A*cycle_time;
Rb92.amount += 1.00*PbndKr92/NA*Kr92.A;
PbndKr93 = ld*Kr93.halflife*Kr93.amount*NA/Kr93.A*cycle_time;
CbndKr93 = ld*Kr93.halflife*Kr93.amount*NA/Kr93.A*cycle_time;
Rb93.amount += 1.00*PbndKr93/NA*Kr93.A;
PbndKr94 = ld*Kr94.halflife*Kr94.amount*NA/Kr94.A*cycle_time;
CbndKr94 = ld*Kr94.halflife*Kr94.amount*NA/Kr94.A*cycle_time;
Rb94.amount += 1.00*PbndKr94/NA*Kr94.A;
PbndKr95 = ld*Kr95.halflife*Kr95.amount*NA/Kr95.A*cycle_time;
CbndKr95 = ld*Kr95.halflife*Kr95.amount*NA/Kr95.A*cycle_time;
Rb95.amount += 1.00*PbndKr95/NA*Kr95.A;
PbndKr97 = ld*Kr97.halflife*Kr97.amount*NA/Kr97.A*cycle_time;
CbndKr97 = ld*Kr97.halflife*Kr97.amount*NA/Kr97.A*cycle_time;
Rb97.amount += 1.00*PbndKr97/NA*Kr97.A;
PbndBr82 = ld*Br82.halflife*Br82.amount*NA/Br82.A*cycle_time;
CbndBr82 = ld*Br82.halflife*Br82.amount*NA/Br82.A*cycle_time;
Kr82.amount += 1.00*PbndBr82/NA*Br82.A;
PbndBr83 = ld*Br83.halflife*Br83.amount*NA/Br83.A*cycle_time;
CbndBr83 = ld*Br83.halflife*Br83.amount*NA/Br83.A*cycle_time;
Kr83.amount += 1.00*PbndBr83/NA*Br83.A;
PbndBr84 = ld*Br84.halflife*Br84.amount*NA/Br84.A*cycle_time;
CbndBr84 = ld*Br84.halflife*Br84.amount*NA/Br84.A*cycle_time;
```

```
Kr84.amount += 1.00*PbndBr84/NA*Br84.A;
PbndBr85 = ld*Br85.halflife*Br85.amount*NA/Br85.A*cycle_time;
CbndBr85 = ld*Br85.halflife*Br85.amount*NA/Br85.A*cycle_time;
PbndBr86 = ld*Br86.halflife*Br86.amount*NA/Br86.A*cycle_time;
CbndBr86 = ld*Br86.halflife*Br86.amount*NA/Br86.A*cycle_time;
PbndBr87 = ld*Br87.halflife*Br87.amount*NA/Br87.A*cycle_time;
CbndBr87 = ld*Br87.halflife*Br87.amount*NA/Br87.A*cycle_time;
Kr87.amount += 1.00*PbndBr87/NA*Br87.A;
PbndBr88 = ld*Br88.halflife*Br88.amount*NA/Br88.A*cycle_time;
CbndBr88 = ld*Br88.halflife*Br88.amount*NA/Br88.A*cycle_time;
Kr88.amount += 1.00*PbndBr88/NA*Br88.A;
PbndBr89 = ld*Br89.halflife*Br89.amount*NA/Br89.A*cycle_time;
CbndBr89 = ld*Br89.halflife*Br89.amount*NA/Br89.A*cycle_time;
Kr89.amount += 1.00*PbndBr89/NA*Br89.A;
PbndBr90 = ld*Br90.halflife*Br90.amount*NA/Br90.A*cycle_time;
CbndBr90 = ld*Br90.halflife*Br90.amount*NA/Br90.A*cycle_time;
Kr90.amount += 1.00*PbndBr90/NA*Br90.A;
PbndBr91 = ld*Br91.halflife*Br91.amount*NA/Br91.A*cycle_time;
CbndBr91 = ld*Br91.halflife*Br91.amount*NA/Br91.A*cycle_time;
Kr91.amount += 1.00*PbndBr91/NA*Br91.A;
PbndBr92 = ld*Br92.halflife*Br92.amount*NA/Br92.A*cycle_time;
CbndBr92 = ld*Br92.halflife*Br92.amount*NA/Br92.A*cycle_time;
Kr92.amount += 1.00*PbndBr92/NA*Br92.A;
PbndSe81 = ld*Se81.halflife*Se81.amount*NA/Se81.A*cycle_time;
CbndSe81 = ld*Se81.halflife*Se81.amount*NA/Se81.A*cycle_time;
Br81.amount += 1.00*PbndSe81/NA*Se81.A;
PbndSe83 = ld*Se83.halflife*Se83.amount*NA/Se83.A*cycle_time;
CbndSe83 = ld*Se83.halflife*Se83.amount*NA/Se83.A*cycle_time;
Br83.amount += 1.00*PbndSe83/NA*Se83.A;
PbndSe84 = ld*Se84.halflife*Se84.amount*NA/Se84.A*cycle_time;
CbndSe84 = ld*Se84.halflife*Se84.amount*NA/Se84.A*cycle_time;
Br84.amount += 1.00*PbndSe84/NA*Se84.A;
```

```
PbndSe85 = ld*Se85.halflife*Se85.amount*NA/Se85.A*cycle_time;
CbndSe85 = ld*Se85.halflife*Se85.amount*NA/Se85.A*cycle_time;
Br85.amount += 1.00*PbndSe85/NA*Se85.A;
PbndSe86 = ld*Se86.halflife*Se86.amount*NA/Se86.A*cycle_time;
CbndSe86 = ld*Se86.halflife*Se86.amount*NA/Se86.A*cycle_time;
Br86.amount += 1.00*PbndSe86/NA*Se86.A;
PbndSe87 = ld*Se87.halflife*Se87.amount*NA/Se87.A*cycle_time;
CbndSe87 = ld*Se87.halflife*Se87.amount*NA/Se87.A*cycle_time;
Br87.amount += 1.00*PbndSe87/NA*Se87.A;
PbndSe88 = ld*Se88.halflife*Se88.amount*NA/Se88.A*cycle_time;
CbndSe88 = ld*Se88.halflife*Se88.amount*NA/Se88.A*cycle_time;
Br88.amount += 1.00*PbndSe88/NA*Se88.A;
PbndSe89 = ld*Se89.halflife*Se89.amount*NA/Se89.A*cycle_time;
CbndSe89 = ld*Se89.halflife*Se89.amount*NA/Se89.A*cycle_time;
Br89.amount += 1.00*PbndSe89/NA*Se89.A;
PbndSe91 = ld*Se91.halflife*Se91.amount*NA/Se91.A*cycle_time;
CbndSe91 = ld*Se91.halflife*Se91.amount*NA/Se91.A*cycle_time;
Br91.amount += 1.00*PbndSe91/NA*Se91.A;
PbndAs77 = ld*As77.halflife*As77.amount*NA/As77.A*cycle_time;
CbndAs77 = ld*As77.halflife*As77.amount*NA/As77.A*cycle_time;
Se77.amount += 1.00*PbndAs77/NA*As77.A;
PbndAs78 = ld*As78.halflife*As78.amount*NA/As78.A*cycle_time;
CbndAs78 = ld*As78.halflife*As78.amount*NA/As78.A*cycle_time;
Se78.amount += 1.00*PbndAs78/NA*As78.A;
PbndAs79 = ld*As79.halflife*As79.amount*NA/As79.A*cycle_time;
CbndAs79 = ld*As79.halflife*As79.amount*NA/As79.A*cycle_time;
Se79.amount += 1.00*PbndAs79/NA*As79.A;
PbndAs80 = ld*As80.halflife*As80.amount*NA/As80.A*cycle_time;
CbndAs80 = ld*As80.halflife*As80.amount*NA/As80.A*cycle_time;
Se80.amount += 1.00*PbndAs80/NA*As80.A;
PbndAs81 = ld*As81.halflife*As81.amount*NA/As81.A*cycle_time;
CbndAs81 = ld*As81.halflife*As81.amount*NA/As81.A*cycle_time;
```

```
Se81.amount += 1.00*PbndAs81/NA*As81.A;
PbndAs82 = ld*As82.halflife*As82.amount*NA/As82.A*cycle_time;
CbndAs82 = ld*As82.halflife*As82.amount*NA/As82.A*cycle_time;
Se82.amount += 1.00*PbndAs82/NA*As82.A;
PbndAs83 = ld*As83.halflife*As83.amount*NA/As83.A*cycle_time;
CbndAs83 = ld*As83.halflife*As83.amount*NA/As83.A*cycle_time;
Se83.amount += 1.00*PbndAs83/NA*As83.A;
PbndAs84 = ld*As84.halflife*As84.amount*NA/As84.A*cycle_time;
CbndAs84 = ld*As84.halflife*As84.amount*NA/As84.A*cycle_time;
Se84.amount += 1.00*PbndAs84/NA*As84.A;
PbndAs85 = ld*As85.halflife*As85.amount*NA/As85.A*cycle_time;
CbndAs85 = ld*As85.halflife*As85.amount*NA/As85.A*cycle_time;
Se85.amount += 1.00*PbndAs85/NA*As85.A;
PbndAs86 = ld*As86.halflife*As86.amount*NA/As86.A*cycle_time;
CbndAs86 = ld*As86.halflife*As86.amount*NA/As86.A*cycle_time;
Se86.amount += 1.00*PbndAs86/NA*As86.A;
PbndAs87 = ld*As87.halflife*As87.amount*NA/As87.A*cycle_time;
CbndAs87 = ld*As87.halflife*As87.amount*NA/As87.A*cycle_time;
Se87.amount += 1.00*PbndAs87/NA*As87.A;
PbndGe75 = ld*Ge75.halflife*Ge75.amount*NA/Ge75.A*cycle_time;
CbndGe75 = ld*Ge75.halflife*Ge75.amount*NA/Ge75.A*cycle_time;
As75.amount += 1.00*PbndGe75/NA*Ge75.A;
PbndGe77 = ld*Ge77.halflife*Ge77.amount*NA/Ge77.A*cycle_time;
CbndGe77 = ld*Ge77.halflife*Ge77.amount*NA/Ge77.A*cycle_time;
As77.amount += 1.00*PbndGe77/NA*Ge77.A;
PbndGe78 = ld*Ge78.halflife*Ge78.amount*NA/Ge78.A*cycle_time;
CbndGe78 = ld*Ge78.halflife*Ge78.amount*NA/Ge78.A*cycle_time;
As78.amount += 1.00*PbndGe78/NA*Ge78.A;
PbndGe79 = ld*Ge79.halflife*Ge79.amount*NA/Ge79.A*cycle_time;
CbndGe79 = ld*Ge79.halflife*Ge79.amount*NA/Ge79.A*cycle_time;
As79.amount += 1.00*PbndGe79/NA*Ge79.A;
PbndGe80 = ld*Ge80.halflife*Ge80.amount*NA/Ge80.A*cycle_time;
```

```
CbndGe80 = ld*Ge80.halflife*Ge80.amount*NA/Ge80.A*cycle_time;
As80.amount += 1.00*PbndGe80/NA*Ge80.A;
PbndGe81 = ld*Ge81.halflife*Ge81.amount*NA/Ge81.A*cycle_time;
CbndGe81 = ld*Ge81.halflife*Ge81.amount*NA/Ge81.A*cycle_time;
As81.amount += 1.00*PbndGe81/NA*Ge81.A;
PbndGe82 = ld*Ge82.halflife*Ge82.amount*NA/Ge82.A*cycle_time;
CbndGe82 = ld*Ge82.halflife*Ge82.amount*NA/Ge82.A*cycle_time;
As82.amount += 1.00*PbndGe82/NA*Ge82.A;
PbndGe83 = ld*Ge83.halflife*Ge83.amount*NA/Ge83.A*cycle_time;
CbndGe83 = ld*Ge83.halflife*Ge83.amount*NA/Ge83.A*cycle_time;
As83.amount += 1.00*PbndGe83/NA*Ge83.A;
PbndGe84 = ld*Ge84.halflife*Ge84.amount*NA/Ge84.A*cycle_time;
CbndGe84 = ld*Ge84.halflife*Ge84.amount*NA/Ge84.A*cycle_time;
As84.amount += 1.00*PbndGe84/NA*Ge84.A;
PbndGa72 = ld*Ga72.halflife*Ga72.amount*NA/Ga72.A*cycle_time;
CbndGa72 = ld*Ga72.halflife*Ga72.amount*NA/Ga72.A*cycle_time;
Ge72.amount += 1.00*PbndGa72/NA*Ga72.A;
PbndGa73 = ld*Ga73.halflife*Ga73.amount*NA/Ga73.A*cycle_time;
CbndGa73 = ld*Ga73.halflife*Ga73.amount*NA/Ga73.A*cycle_time;
Ge73.amount += 1.00*PbndGa73/NA*Ga73.A;
PbndGa74 = ld*Ga74.halflife*Ga74.amount*NA/Ga74.A*cycle_time;
CbndGa74 = ld*Ga74.halflife*Ga74.amount*NA/Ga74.A*cycle_time;
Ge74.amount += 1.00*PbndGa74/NA*Ga74.A;
PbndGa75 = ld*Ga75.halflife*Ga75.amount*NA/Ga75.A*cycle_time;
CbndGa75 = ld*Ga75.halflife*Ga75.amount*NA/Ga75.A*cycle_time;
Ge75.amount += 1.00*PbndGa75/NA*Ga75.A;
PbndGa76 = ld*Ga76.halflife*Ga76.amount*NA/Ga76.A*cycle_time;
CbndGa76 = ld*Ga76.halflife*Ga76.amount*NA/Ga76.A*cycle_time;
Ge76.amount += 1.00*PbndGa76/NA*Ga76.A;
PbndGa77 = ld*Ga77.halflife*Ga77.amount*NA/Ga77.A*cycle_time;
CbndGa77 = ld*Ga77.halflife*Ga77.amount*NA/Ga77.A*cycle_time;
Ge77.amount += 1.00*PbndGa77/NA*Ga77.A;
```

```
PbndGa78 = ld*Ga78.halflife*Ga78.amount*NA/Ga78.A*cycle_time;
CbndGa78 = ld*Ga78.halflife*Ga78.amount*NA/Ga78.A*cycle_time;
Ge78.amount += 1.00*PbndGa78/NA*Ga78.A;
PbndGa79 = ld*Ga79.halflife*Ga79.amount*NA/Ga79.A*cycle_time;
CbndGa79 = ld*Ga79.halflife*Ga79.amount*NA/Ga79.A*cycle_time;
Ge79.amount += 1.00*PbndGa79/NA*Ga79.A;
PbndGa80 = ld*Ga80.halflife*Ga80.amount*NA/Ga80.A*cycle_time;
CbndGa80 = ld*Ga80.halflife*Ga80.amount*NA/Ga80.A*cycle_time;
Ge80.amount += 1.00*PbndGa80/NA*Ga80.A;
PbndGa81 = ld*Ga81.halflife*Ga81.amount*NA/Ga81.A*cycle_time;
CbndGa81 = ld*Ga81.halflife*Ga81.amount*NA/Ga81.A*cycle_time;
Ge81.amount += 1.00*PbndGa81/NA*Ga81.A;
PbndGa82 = ld*Ga82.halflife*Ga82.amount*NA/Ga82.A*cycle_time;
CbndGa82 = ld*Ga82.halflife*Ga82.amount*NA/Ga82.A*cycle_time;
Ge82.amount += 1.00*PbndGa82/NA*Ga82.A;
PbndGa83 = ld*Ga83.halflife*Ga83.amount*NA/Ga83.A*cycle_time;
CbndGa83 = ld*Ga83.halflife*Ga83.amount*NA/Ga83.A*cycle_time;
Ge83.amount += 1.00*PbndGa83/NA*Ga83.A;
PbndZn72 = ld*Zn72.halflife*Zn72.amount*NA/Zn72.A*cycle_time;
CbndZn72 = ld*Zn72.halflife*Zn72.amount*NA/Zn72.A*cycle_time;
Ga72.amount += 1.00*PbndZn72/NA*Zn72.A;
PbndZn73 = ld*Zn73.halflife*Zn73.amount*NA/Zn73.A*cycle_time;
CbndZn73 = ld*Zn73.halflife*Zn73.amount*NA/Zn73.A*cycle_time;
Ga73.amount += 1.00*PbndZn73/NA*Zn73.A;
PbndZn74 = ld*Zn74.halflife*Zn74.amount*NA/Zn74.A*cycle_time;
CbndZn74 = ld*Zn74.halflife*Zn74.amount*NA/Zn74.A*cycle_time;
Ga74.amount += 1.00*PbndZn74/NA*Zn74.A;
PbndZn75 = ld*Zn75.halflife*Zn75.amount*NA/Zn75.A*cycle_time;
CbndZn75 = ld*Zn75.halflife*Zn75.amount*NA/Zn75.A*cycle_time;
Ga75.amount += 1.00*PbndZn75/NA*Zn75.A;
PbndZn76 = ld*Zn76.halflife*Zn76.amount*NA/Zn76.A*cycle_time;
CbndZn76 = ld*Zn76.halflife*Zn76.amount*NA/Zn76.A*cycle_time;
```

```
Ga76.amount += 1.00*PbndZn76/NA*Zn76.A;
PbndZn77 = ld*Zn77.halflife*Zn77.amount*NA/Zn77.A*cycle_time;
CbndZn77 = ld*Zn77.halflife*Zn77.amount*NA/Zn77.A*cycle_time;
Ga77.amount += 1.00*PbndZn77/NA*Zn77.A;
PbndZn78 = ld*Zn78.halflife*Zn78.amount*NA/Zn78.A*cycle_time;
CbndZn78 = ld*Zn78.halflife*Zn78.amount*NA/Zn78.A*cycle_time;
Ga78.amount += 1.00*PbndZn78/NA*Zn78.A;
PbndZn79 = ld*Zn79.halflife*Zn79.amount*NA/Zn79.A*cycle_time;
CbndZn79 = ld*Zn79.halflife*Zn79.amount*NA/Zn79.A*cycle_time;
Ga79.amount += 1.00*PbndZn79/NA*Zn79.A;
PbndZn80 = ld*Zn80.halflife*Zn80.amount*NA/Zn80.A*cycle_time;
CbndZn80 = ld*Zn80.halflife*Zn80.amount*NA/Zn80.A*cycle_time;
Ga80.amount += 1.00*PbndZn80/NA*Zn80.A;
PbndCu72 = ld*Cu72.halflife*Cu72.amount*NA/Cu72.A*cycle_time;
CbndCu72 = ld*Cu72.halflife*Cu72.amount*NA/Cu72.A*cycle_time;
Zn72.amount += 1.00*PbndCu72/NA*Cu72.A;
PbndCu73 = ld*Cu73.halflife*Cu73.amount*NA/Cu73.A*cycle_time;
CbndCu73 = ld*Cu73.halflife*Cu73.amount*NA/Cu73.A*cycle_time;
Zn73.amount += 1.00*PbndCu73/NA*Cu73.A;
PbndCu75 = ld*Cu75.halflife*Cu75.amount*NA/Cu75.A*cycle_time;
CbndCu75 = ld*Cu75.halflife*Cu75.amount*NA/Cu75.A*cycle_time;
Zn75.amount += 1.00*PbndCu75/NA*Cu75.A;
PbndCu76 = ld*Cu76.halflife*Cu76.amount*NA/Cu76.A*cycle_time;
CbndCu76 = ld*Cu76.halflife*Cu76.amount*NA/Cu76.A*cycle_time;
Zn76.amount += 1.00*PbndCu76/NA*Cu76.A;
```

Appendix VI

Probabilistic space equations of the sorted isobars

This appendix shows the probabilistic space equations for isobars generated at the end of first cycle for the Experiment No.2.

```
Ag109.amount += distr_A1U235[110]*pdf_dZ[4]/NA*110;
Ag110.amount += distr_A1U235[111]*pdf_dZ[4]/NA*111;
Ag111.amount += distr_A1U235[112]*pdf_dZ[3]/NA*112;
Ag112.amount += distr_A1U235[113]*pdf_dZ[3]/NA*113;
Ag113.amount += distr_A1U235[114]*pdf_dZ[3]/NA*114;
Ag114.amount += distr_A1U235[115]*pdf_dZ[2]/NA*115;
Ag115.amount += distr_A1U235[116]*pdf_dZ[2]/NA*116;
Ag116.amount += distr_A1U235[117]*pdf_dZ[1]/NA*117;
Ag117.amount += distr_A2U235[118]*pdf_dZ[1]/NA*118;
Ag118.amount += distr_A2U235[119]*pdf_dZ[1]/NA*119;
Ag119.amount += distr_A2U235[120]*pdf_dZ[0]/NA*120;
Ag120.amount += distr_A2U235[121]*pdf_dZ[0]/NA*121;
Ag121.amount += distr_A2U235[122]*pdf_dZ[-1]/NA*122;
Ag122.amount += distr_A2U235[123]*pdf_dZ[-1]/NA*123;
Ag123.amount += distr_A2U235[124]*pdf_dZ[-1]/NA*124;
Ag124.amount += distr_A2U235[125]*pdf_dZ[-2]/NA*125;
Ag125.amount += distr_A2U235[126]*pdf_dZ[-2]/NA*126;
Ag126.amount += distr_A2U235[127]*pdf_dZ[-3]/NA*127;
Ag127.amount += distr_A2U235[128]*pdf_dZ[-3]/NA*128;
Ag128.amount += distr_A2U235[129]*pdf_dZ[-3]/NA*129;
Ag129.amount += distr_A2U235[130]*pdf_dZ[-4]/NA*130;
Ag130.amount += distr_A2U235[131]*pdf_dZ[-4]/NA*131;
Ag131.amount += distr_A2U235[132]*pdf_dZ[-4]/NA*132;
As73.amount += distr_A1U235[74]*pdf_dZ[4]/NA*74;
As74.amount += distr_A1U235[75]*pdf_dZ[4]/NA*75;
As75.amount += distr_A1U235[76]*pdf_dZ[4]/NA*76;
As76.amount += distr_A1U235[77]*pdf_dZ[3]/NA*77;
As77.amount += distr_A1U235[78]*pdf_dZ[3]/NA*78;
As78.amount += distr_A1U235[79]*pdf_dZ[2]/NA*79;
As79.amount += distr_A1U235[80]*pdf_dZ[2]/NA*80;
As80.amount += distr_A1U235[81]*pdf_dZ[2]/NA*81;
As81.amount += distr_A1U235[82]*pdf_dZ[1]/NA*82;
```

```
As82.amount += distr_A1U235[83]*pdf_dZ[1]/NA*83;
As83.amount += distr_A1U235[84]*pdf_dZ[0]/NA*84;
As84.amount += distr_A1U235[85]*pdf_dZ[0]/NA*85;
As85.amount += distr_A1U235[86]*pdf_dZ[0]/NA*86;
As86.amount += distr_A1U235[87]*pdf_dZ[-1]/NA*87;
As87.amount += distr_A1U235[88]*pdf_dZ[-1]/NA*88;
As88.amount += distr_A1U235[89]*pdf_dZ[-2]/NA*89;
As89.amount += distr_A1U235[90]*pdf_dZ[-2]/NA*90;
As90.amount += distr_A1U235[91]*pdf_dZ[-2]/NA*91;
As91.amount += distr_A1U235[92]*pdf_dZ[-3]/NA*92;
As92.amount += distr_A1U235[93]*pdf_dZ[-3]/NA*93;
As93.amount += distr_A1U235[94]*pdf_dZ[-4]/NA*94;
As94.amount += distr_A1U235[95]*pdf_dZ[-4]/NA*95;
As95.amount += distr_A1U235[96]*pdf_dZ[-4]/NA*96;
Ba132.amount += distr_A2U235[133]*pdf_dZ[4]/NA*133;
Ba133.amount += distr_A2U235[134]*pdf_dZ[4]/NA*134;
Ba134.amount += distr_A2U235[135]*pdf_dZ[3]/NA*135;
Ba135.amount += distr_A2U235[136]*pdf_dZ[3]/NA*136;
Ba136.amount += distr_A2U235[137]*pdf_dZ[3]/NA*137;
Ba137.amount += distr_A2U235[138]*pdf_dZ[2]/NA*138;
Ba138.amount += distr_A2U235[139]*pdf_dZ[2]/NA*139;
Ba139.amount += distr_A2U235[140]*pdf_dZ[1]/NA*140;
Ba140.amount += distr_A2U235[141]*pdf_dZ[1]/NA*141;
Ba141.amount += distr_A2U235[142]*pdf_dZ[1]/NA*142;
Ba142.amount += distr_A2U235[143]*pdf_dZ[0]/NA*143;
Ba143.amount += distr_A2U235[144]*pdf_dZ[0]/NA*144;
Ba144.amount += distr_A2U235[145]*pdf_dZ[-1]/NA*145;
Ba145.amount += distr_A2U235[146]*pdf_dZ[-1]/NA*146;
Ba146.amount += distr_A2U235[147]*pdf_dZ[-1]/NA*147;
Ba147.amount += distr_A2U235[148]*pdf_dZ[-2]/NA*148;
Ba148.amount += distr_A2U235[149]*pdf_dZ[-2]/NA*149;
Ba149.amount += distr_A2U235[150]*pdf_dZ[-3]/NA*150;
```

```
Ba150.amount += distr_A2U235[151]*pdf_dZ[-3]/NA*151;
Ba151.amount += distr_A2U235[152]*pdf_dZ[-3]/NA*152;
Ba152.amount += distr_A2U235[153]*pdf_dZ[-4]/NA*153;
Ba153.amount += distr_A2U235[154]*pdf_dZ[-4]/NA*154;
Br100.amount += distr_A1U235[101]*pdf_dZ[-4]/NA*101;
Br78.amount += distr_A1U235[79]*pdf_dZ[4]/NA*79;
Br79.amount += distr_A1U235[80]*pdf_dZ[4]/NA*80;
Br80.amount += distr_A1U235[81]*pdf_dZ[4]/NA*81;
Br81.amount += distr_A1U235[82]*pdf_dZ[3]/NA*82;
Br82.amount += distr_A1U235[83]*pdf_dZ[3]/NA*83;
Br83.amount += distr_A1U235[84]*pdf_dZ[2]/NA*84;
Br84.amount += distr_A1U235[85]*pdf_dZ[2]/NA*85;
Br85.amount += distr_A1U235[86]*pdf_dZ[2]/NA*86;
Br86.amount += distr_A1U235[87]*pdf_dZ[1]/NA*87;
Br87.amount += distr_A1U235[88]*pdf_dZ[1]/NA*88;
Br88.amount += distr_A1U235[89]*pdf_dZ[0]/NA*89;
Br89.amount += distr_A1U235[90]*pdf_dZ[0]/NA*90;
Br90.amount += distr_A1U235[91]*pdf_dZ[0]/NA*91;
Br91.amount += distr_A1U235[92]*pdf_dZ[-1]/NA*92;
Br92.amount += distr_A1U235[93]*pdf_dZ[-1]/NA*93;
Br93.amount += distr_A1U235[94]*pdf_dZ[-2]/NA*94;
Br94.amount += distr_A1U235[95]*pdf_dZ[-2]/NA*95;
Br95.amount += distr_A1U235[96]*pdf_dZ[-2]/NA*96;
Br96.amount += distr_A1U235[97]*pdf_dZ[-3]/NA*97;
Br97.amount += distr_A1U235[98]*pdf_dZ[-3]/NA*98;
Br98.amount += distr_A1U235[99]*pdf_dZ[-3]/NA*99;
Br99.amount += distr_A1U235[100]*pdf_dZ[-4]/NA*100;
Cd111.amount += distr_A1U235[112]*pdf_dZ[4]/NA*112;
Cd112.amount += distr_A1U235[113]*pdf_dZ[4]/NA*113;
Cd113.amount += distr_A1U235[114]*pdf_dZ[4]/NA*114;
Cd114.amount += distr_A1U235[115]*pdf_dZ[3]/NA*115;
Cd115.amount += distr_A1U235[116]*pdf_dZ[3]/NA*116;
```

```
Cd116.amount += distr_A1U235[117]*pdf_dZ[2]/NA*117;
Cd117.amount += distr_A2U235[118]*pdf_dZ[2]/NA*118;
Cd118.amount += distr_A2U235[119]*pdf_dZ[2]/NA*119;
Cd119.amount += distr_A2U235[120]*pdf_dZ[1]/NA*120;
Cd120.amount += distr_A2U235[121]*pdf_dZ[1]/NA*121;
Cd121.amount += distr_A2U235[122]*pdf_dZ[0]/NA*122;
Cd122.amount += distr_A2U235[123]*pdf_dZ[0]/NA*123;
Cd123.amount += distr_A2U235[124]*pdf_dZ[0]/NA*124;
Cd124.amount += distr_A2U235[125]*pdf_dZ[-1]/NA*125;
Cd125.amount += distr_A2U235[126]*pdf_dZ[-1]/NA*126;
Cd126.amount += distr_A2U235[127]*pdf_dZ[-2]/NA*127;
Cd127.amount += distr_A2U235[128]*pdf_dZ[-2]/NA*128;
Cd128.amount += distr_A2U235[129]*pdf_dZ[-2]/NA*129;
Cd129.amount += distr_A2U235[130]*pdf_dZ[-3]/NA*130;
Cd130.amount += distr_A2U235[131]*pdf_dZ[-3]/NA*131;
Cd131.amount += distr_A2U235[132]*pdf_dZ[-3]/NA*132;
Cd132.amount += distr_A2U235[133]*pdf_dZ[-4]/NA*133;
Cd133.amount += distr_A2U235[134]*pdf_dZ[-4]/NA*134;
Ce137.amount += distr_A2U235[138]*pdf_dZ[4]/NA*138;
Ce138.amount += distr_A2U235[139]*pdf_dZ[4]/NA*139;
Ce139.amount += distr_A2U235[140]*pdf_dZ[3]/NA*140;
Ce140.amount += distr_A2U235[141]*pdf_dZ[3]/NA*141;
Ce141.amount += distr_A2U235[142]*pdf_dZ[3]/NA*142;
Ce142.amount += distr_A2U235[143]*pdf_dZ[2]/NA*143;
Ce143.amount += distr_A2U235[144]*pdf_dZ[2]/NA*144;
Ce144.amount += distr_A2U235[145]*pdf_dZ[1]/NA*145;
Ce145.amount += distr_A2U235[146]*pdf_dZ[1]/NA*146;
Ce146.amount += distr_A2U235[147]*pdf_dZ[1]/NA*147;
Ce147.amount += distr_A2U235[148]*pdf_dZ[0]/NA*148;
Ce148.amount += distr_A2U235[149]*pdf_dZ[0]/NA*149;
Ce149.amount += distr_A2U235[150]*pdf_dZ[-1]/NA*150;
Ce150.amount += distr_A2U235[151]*pdf_dZ[-1]/NA*151;
```

```
Ce151.amount += distr_A2U235[152]*pdf_dZ[-1]/NA*152;
Ce152.amount += distr_A2U235[153]*pdf_dZ[-2]/NA*153;
Ce153.amount += distr_A2U235[154]*pdf_dZ[-2]/NA*154;
Ce154.amount += distr_A2U235[155]*pdf_dZ[-3]/NA*155;
Ce155.amount += distr_A2U235[156]*pdf_dZ[-3]/NA*156;
Ce156.amount += distr_A2U235[157]*pdf_dZ[-3]/NA*157;
Ce157.amount += distr_A2U235[158]*pdf_dZ[-4]/NA*158;
Ce158.amount += distr_A2U235[159]*pdf_dZ[-4]/NA*159;
Co72.amount += distr_A1U235[73]*pdf_dZ[-1]/NA*73;
Co73.amount += distr_A1U235[74]*pdf_dZ[-2]/NA*74;
Co74.amount += distr_A1U235[75]*pdf_dZ[-2]/NA*75;
Co75.amount += distr_A1U235[76]*pdf_dZ[-2]/NA*76;
Co76.amount += distr_A1U235[77]*pdf_dZ[-3]/NA*77;
Co77.amount += distr_A1U235[78]*pdf_dZ[-3]/NA*78;
Co78.amount += distr_A1U235[79]*pdf_dZ[-4]/NA*79;
Co79.amount += distr_A1U235[80]*pdf_dZ[-4]/NA*80;
Co80.amount += distr_A1U235[81]*pdf_dZ[-4]/NA*81;
Cr72.amount += distr_A1U235[73]*pdf_dZ[-4]/NA*73;
Cs129.amount += distr_A2U235[130]*pdf_dZ[4]/NA*130;
Cs130.amount += distr_A2U235[131]*pdf_dZ[4]/NA*131;
Cs131.amount += distr_A2U235[132]*pdf_dZ[4]/NA*132;
Cs132.amount += distr_A2U235[133]*pdf_dZ[3]/NA*133;
Cs133.amount += distr_A2U235[134]*pdf_dZ[3]/NA*134;
Cs134.amount += distr_A2U235[135]*pdf_dZ[2]/NA*135;
Cs135.amount += distr_A2U235[136]*pdf_dZ[2]/NA*136;
Cs136.amount += distr_A2U235[137]*pdf_dZ[2]/NA*137;
Cs137.amount += distr_A2U235[138]*pdf_dZ[1]/NA*138;
Cs138.amount += distr_A2U235[139]*pdf_dZ[1]/NA*139;
Cs139.amount += distr_A2U235[140]*pdf_dZ[0]/NA*140;
Cs140.amount += distr_A2U235[141]*pdf_dZ[0]/NA*141;
Cs141.amount += distr_A2U235[142]*pdf_dZ[0]/NA*142;
Cs142.amount += distr_A2U235[143]*pdf_dZ[-1]/NA*143;
```

```
Cs143.amount += distr_A2U235[144]*pdf_dZ[-1]/NA*144;
Cs144.amount += distr_A2U235[145]*pdf_dZ[-2]/NA*145;
Cs145.amount += distr_A2U235[146]*pdf_dZ[-2]/NA*146;
Cs146.amount += distr_A2U235[147]*pdf_dZ[-2]/NA*147;
Cs147.amount += distr_A2U235[148]*pdf_dZ[-3]/NA*148;
Cs148.amount += distr_A2U235[149]*pdf_dZ[-3]/NA*149;
Cs149.amount += distr_A2U235[150]*pdf_dZ[-4]/NA*150;
Cs150.amount += distr_A2U235[151]*pdf_dZ[-4]/NA*151;
Cs151.amount += distr_A2U235[152]*pdf_dZ[-4]/NA*152;
Cu72.amount += distr_A1U235[73]*pdf_dZ[1]/NA*73;
Cu73.amount += distr_A1U235[74]*pdf_dZ[0]/NA*74;
Cu74.amount += distr_A1U235[75]*pdf_dZ[0]/NA*75;
Cu75.amount += distr_A1U235[76]*pdf_dZ[0]/NA*76;
Cu76.amount += distr_A1U235[77]*pdf_dZ[-1]/NA*77;
Cu77.amount += distr_A1U235[78]*pdf_dZ[-1]/NA*78;
Cu78.amount += distr_A1U235[79]*pdf_dZ[-2]/NA*79;
Cu79.amount += distr_A1U235[80]*pdf_dZ[-2]/NA*80;
Cu80.amount += distr_A1U235[81]*pdf_dZ[-2]/NA*81;
Cu81.amount += distr_A1U235[82]*pdf_dZ[-3]/NA*82;
Cu82.amount += distr_A1U235[83]*pdf_dZ[-3]/NA*83;
Cu83.amount += distr_A1U235[84]*pdf_dZ[-4]/NA*84;
Cu84.amount += distr_A1U235[85]*pdf_dZ[-4]/NA*85;
Cu85.amount += distr_A1U235[86]*pdf_dZ[-4]/NA*86;
Dy157.amount += distr_A2U235[158]*pdf_dZ[4]/NA*158;
Dy158.amount += distr_A2U235[159]*pdf_dZ[4]/NA*159;
Dy159.amount += distr_A2U235[160]*pdf_dZ[3]/NA*160;
Dy160.amount += distr_A2U235[161]*pdf_dZ[3]/NA*161;
Dy161.amount += distr_A2U235[162]*pdf_dZ[3]/NA*162;
Eu149.amount += distr_A2U235[150]*pdf_dZ[4]/NA*150;
Eu150.amount += distr_A2U235[151]*pdf_dZ[4]/NA*151;
Eu151.amount += distr_A2U235[152]*pdf_dZ[4]/NA*152;
Eu152.amount += distr_A2U235[153]*pdf_dZ[3]/NA*153;
```

```
Eu153.amount += distr_A2U235[154]*pdf_dZ[3]/NA*154;
Eu154.amount += distr_A2U235[155]*pdf_dZ[2]/NA*155;
Eu155.amount += distr_A2U235[156]*pdf_dZ[2]/NA*156;
Eu156.amount += distr_A2U235[157]*pdf_dZ[2]/NA*157;
Eu157.amount += distr_A2U235[158]*pdf_dZ[1]/NA*158;
Eu158.amount += distr_A2U235[159]*pdf_dZ[1]/NA*159;
Eu159.amount += distr_A2U235[160]*pdf_dZ[0]/NA*160;
Eu160.amount += distr_A2U235[161]*pdf_dZ[0]/NA*161;
Eu161.amount += distr_A2U235[162]*pdf_dZ[0]/NA*162;
Fe72.amount += distr_A1U235[73]*pdf_dZ[-2]/NA*73;
Fe73.amount += distr_A1U235[74]*pdf_dZ[-3]/NA*74;
Fe74.amount += distr_A1U235[75]*pdf_dZ[-3]/NA*75;
Fe75.amount += distr_A1U235[76]*pdf_dZ[-3]/NA*76;
Fe76.amount += distr_A1U235[77]*pdf_dZ[-4]/NA*77;
Fe77.amount += distr_A1U235[78]*pdf_dZ[-4]/NA*78;
Ga72.amount += distr_A1U235[73]*pdf_dZ[3]/NA*73;
Ga73.amount += distr_A1U235[74]*pdf_dZ[2]/NA*74;
Ga74.amount += distr_A1U235[75]*pdf_dZ[2]/NA*75;
Ga75.amount += distr_A1U235[76]*pdf_dZ[2]/NA*76;
Ga76.amount += distr_A1U235[77]*pdf_dZ[1]/NA*77;
Ga77.amount += distr_A1U235[78]*pdf_dZ[1]/NA*78;
Ga78.amount += distr_A1U235[79]*pdf_dZ[0]/NA*79;
Ga79.amount += distr_A1U235[80]*pdf_dZ[0]/NA*80;
Ga80.amount += distr_A1U235[81]*pdf_dZ[0]/NA*81;
Ga81.amount += distr_A1U235[82]*pdf_dZ[-1]/NA*82;
Ga82.amount += distr_A1U235[83]*pdf_dZ[-1]/NA*83;
Ga83.amount += distr_A1U235[84]*pdf_dZ[-2]/NA*84;
Ga84.amount += distr_A1U235[85]*pdf_dZ[-2]/NA*85;
Ga85.amount += distr_A1U235[86]*pdf_dZ[-2]/NA*86;
Ga86.amount += distr_A1U235[87]*pdf_dZ[-3]/NA*87;
Ga87.amount += distr_A1U235[88]*pdf_dZ[-3]/NA*88;
Ga88.amount += distr_A1U235[89]*pdf_dZ[-4]/NA*89;
```

```
Ga89.amount += distr_A1U235[90]*pdf_dZ[-4]/NA*90;
Ga90.amount += distr_A1U235[91]*pdf_dZ[-4]/NA*91;
Gd152.amount += distr_A2U235[153]*pdf_dZ[4]/NA*153;
Gd153.amount += distr_A2U235[154]*pdf_dZ[4]/NA*154;
Gd154.amount += distr_A2U235[155]*pdf_dZ[3]/NA*155;
Gd155.amount += distr_A2U235[156]*pdf_dZ[3]/NA*156;
Gd156.amount += distr_A2U235[157]*pdf_dZ[3]/NA*157;
Gd157.amount += distr_A2U235[158]*pdf_dZ[2]/NA*158;
Gd158.amount += distr_A2U235[159]*pdf_dZ[2]/NA*159;
Gd159.amount += distr_A2U235[160]*pdf_dZ[1]/NA*160;
Gd160.amount += distr_A2U235[161]*pdf_dZ[1]/NA*161;
Gd161.amount += distr_A2U235[162]*pdf_dZ[1]/NA*162;
Ge72.amount += distr_A1U235[73]*pdf_dZ[4]/NA*73;
Ge73.amount += distr_A1U235[74]*pdf_dZ[3]/NA*74;
Ge74.amount += distr_A1U235[75]*pdf_dZ[3]/NA*75;
Ge75.amount += distr_A1U235[76]*pdf_dZ[3]/NA*76;
Ge76.amount += distr_A1U235[77]*pdf_dZ[2]/NA*77;
Ge77.amount += distr_A1U235[78]*pdf_dZ[2]/NA*78;
Ge78.amount += distr_A1U235[79]*pdf_dZ[1]/NA*79;
Ge79.amount += distr_A1U235[80]*pdf_dZ[1]/NA*80;
Ge80.amount += distr_A1U235[81]*pdf_dZ[1]/NA*81;
Ge81.amount += distr_A1U235[82]*pdf_dZ[0]/NA*82;
Ge82.amount += distr_A1U235[83]*pdf_dZ[0]/NA*83;
Ge83.amount += distr_A1U235[84]*pdf_dZ[-1]/NA*84;
Ge84.amount += distr_A1U235[85]*pdf_dZ[-1]/NA*85;
Ge85.amount += distr_A1U235[86]*pdf_dZ[-1]/NA*86;
Ge86.amount += distr_A1U235[87]*pdf_dZ[-2]/NA*87;
Ge87.amount += distr_A1U235[88]*pdf_dZ[-2]/NA*88;
Ge88.amount += distr_A1U235[89]*pdf_dZ[-3]/NA*89;
Ge89.amount += distr_A1U235[90]*pdf_dZ[-3]/NA*90;
Ge90.amount += distr_A1U235[91]*pdf_dZ[-3]/NA*91;
Ge91.amount += distr_A1U235[92]*pdf_dZ[-4]/NA*92;
```

```
Ge92.amount += distr_A1U235[93]*pdf_dZ[-4]/NA*93;
Ho159.amount += distr_A2U235[160]*pdf_dZ[4]/NA*160;
Ho160.amount += distr_A2U235[161]*pdf_dZ[4]/NA*161;
Ho161.amount += distr_A2U235[162]*pdf_dZ[4]/NA*162;
I124.amount += distr_A2U235[125]*pdf_dZ[4]/NA*125;
I125.amount += distr_A2U235[126]*pdf_dZ[4]/NA*126;
I126.amount += distr_A2U235[127]*pdf_dZ[3]/NA*127;
I127.amount += distr_A2U235[128]*pdf_dZ[3]/NA*128;
I128.amount += distr_A2U235[129]*pdf_dZ[3]/NA*129;
I129.amount += distr_A2U235[130]*pdf_dZ[2]/NA*130;
I130.amount += distr_A2U235[131]*pdf_dZ[2]/NA*131;
I131.amount += distr_A2U235[132]*pdf_dZ[2]/NA*132;
I132.amount += distr_A2U235[133]*pdf_dZ[1]/NA*133;
I133.amount += distr_A2U235[134]*pdf_dZ[1]/NA*134;
I134.amount += distr_A2U235[135]*pdf_dZ[0]/NA*135;
I135.amount += distr_A2U235[136]*pdf_dZ[0]/NA*136;
I136.amount += distr_A2U235[137]*pdf_dZ[0]/NA*137;
I137.amount += distr_A2U235[138]*pdf_dZ[-1]/NA*138;
I138.amount += distr_A2U235[139]*pdf_dZ[-1]/NA*139;
I139.amount += distr_A2U235[140]*pdf_dZ[-2]/NA*140;
I140.amount += distr_A2U235[141]*pdf_dZ[-2]/NA*141;
I141.amount += distr_A2U235[142]*pdf_dZ[-2]/NA*142;
I142.amount += distr_A2U235[143]*pdf_dZ[-3]/NA*143;
I143.amount += distr_A2U235[144]*pdf_dZ[-3]/NA*144;
I144.amount += distr_A2U235[145]*pdf_dZ[-4]/NA*145;
I145.amount += distr_A2U235[146]*pdf_dZ[-4]/NA*146;
I146.amount += distr_A2U235[147]*pdf_dZ[-4]/NA*147;
In114.amount += distr_A1U235[115]*pdf_dZ[4]/NA*115;
In115.amount += distr_A1U235[116]*pdf_dZ[4]/NA*116;
In116.amount += distr_A1U235[117]*pdf_dZ[3]/NA*117;
In117.amount += distr_A2U235[118]*pdf_dZ[3]/NA*118;
In118.amount += distr_A2U235[119]*pdf_dZ[3]/NA*119;
```

```
In119.amount += distr_A2U235[120]*pdf_dZ[2]/NA*120;
In120.amount += distr_A2U235[121]*pdf_dZ[2]/NA*121;
In121.amount += distr_A2U235[122]*pdf_dZ[1]/NA*122;
In122.amount += distr_A2U235[123]*pdf_dZ[1]/NA*123;
In123.amount += distr_A2U235[124]*pdf_dZ[1]/NA*124;
In124.amount += distr_A2U235[125]*pdf_dZ[0]/NA*125;
In125.amount += distr_A2U235[126]*pdf_dZ[0]/NA*126;
In126.amount += distr_A2U235[127]*pdf_dZ[-1]/NA*127;
In127.amount += distr_A2U235[128]*pdf_dZ[-1]/NA*128;
In128.amount += distr_A2U235[129]*pdf_dZ[-1]/NA*129;
In129.amount += distr_A2U235[130]*pdf_dZ[-2]/NA*130;
In130.amount += distr_A2U235[131]*pdf_dZ[-2]/NA*131;
In131.amount += distr_A2U235[132]*pdf_dZ[-2]/NA*132;
In132.amount += distr_A2U235[133]*pdf_dZ[-3]/NA*133;
In133.amount += distr_A2U235[134]*pdf_dZ[-3]/NA*134;
In134.amount += distr_A2U235[135]*pdf_dZ[-4]/NA*135;
In135.amount += distr_A2U235[136]*pdf_dZ[-4]/NA*136;
In136.amount += distr_A2U235[137]*pdf_dZ[-4]/NA*137;
Kr100.amount += distr_A1U235[101]*pdf_dZ[-3]/NA*101;
Kr101.amount += distr_A1U235[102]*pdf_dZ[-4]/NA*102;
Kr102.amount += distr_A1U235[103]*pdf_dZ[-4]/NA*103;
Kr103.amount += distr_A1U235[104]*pdf_dZ[-4]/NA*104;
Kr81.amount += distr_A1U235[82]*pdf_dZ[4]/NA*82;
Kr82.amount += distr_A1U235[83]*pdf_dZ[4]/NA*83;
Kr83.amount += distr_A1U235[84]*pdf_dZ[3]/NA*84;
Kr84.amount += distr_A1U235[85]*pdf_dZ[3]/NA*85;
Kr85.amount += distr_A1U235[86]*pdf_dZ[3]/NA*86;
Kr86.amount += distr_A1U235[87]*pdf_dZ[2]/NA*87;
Kr87.amount += distr_A1U235[88]*pdf_dZ[2]/NA*88;
Kr88.amount += distr_A1U235[89]*pdf_dZ[1]/NA*89;
Kr89.amount += distr_A1U235[90]*pdf_dZ[1]/NA*90;
Kr90.amount += distr_A1U235[91]*pdf_dZ[1]/NA*91;
```

```
Kr91.amount += distr_A1U235[92]*pdf_dZ[0]/NA*92;
Kr92.amount += distr_A1U235[93]*pdf_dZ[0]/NA*93;
Kr93.amount += distr_A1U235[94]*pdf_dZ[-1]/NA*94;
Kr94.amount += distr_A1U235[95]*pdf_dZ[-1]/NA*95;
Kr95.amount += distr_A1U235[96]*pdf_dZ[-1]/NA*96;
Kr96.amount += distr_A1U235[97]*pdf_dZ[-2]/NA*97;
Kr97.amount += distr_A1U235[98]*pdf_dZ[-2]/NA*98;
Kr98.amount += distr_A1U235[99]*pdf_dZ[-2]/NA*99;
Kr99.amount += distr_A1U235[100]*pdf_dZ[-3]/NA*100;
La134.amount += distr_A2U235[135]*pdf_dZ[4]/NA*135;
La135.amount += distr_A2U235[136]*pdf_dZ[4]/NA*136;
La136.amount += distr_A2U235[137]*pdf_dZ[4]/NA*137;
La137.amount += distr_A2U235[138]*pdf_dZ[3]/NA*138;
La138.amount += distr_A2U235[139]*pdf_dZ[3]/NA*139;
La139.amount += distr_A2U235[140]*pdf_dZ[2]/NA*140;
La140.amount += distr_A2U235[141]*pdf_dZ[2]/NA*141;
La141.amount += distr_A2U235[142]*pdf_dZ[2]/NA*142;
La142.amount += distr_A2U235[143]*pdf_dZ[1]/NA*143;
La143.amount += distr_A2U235[144]*pdf_dZ[1]/NA*144;
La144.amount += distr_A2U235[145]*pdf_dZ[0]/NA*145;
La145.amount += distr_A2U235[146]*pdf_dZ[0]/NA*146;
La146.amount += distr_A2U235[147]*pdf_dZ[0]/NA*147;
La147.amount += distr_A2U235[148]*pdf_dZ[-1]/NA*148;
La148.amount += distr_A2U235[149]*pdf_dZ[-1]/NA*149;
La149.amount += distr_A2U235[150]*pdf_dZ[-2]/NA*150;
La150.amount += distr_A2U235[151]*pdf_dZ[-2]/NA*151;
La151.amount += distr_A2U235[152]*pdf_dZ[-2]/NA*152;
La152.amount += distr_A2U235[153]*pdf_dZ[-3]/NA*153;
La153.amount += distr_A2U235[154]*pdf_dZ[-3]/NA*154;
La154.amount += distr_A2U235[155]*pdf_dZ[-4]/NA*155;
La155.amount += distr_A2U235[156]*pdf_dZ[-4]/NA*156;
La156.amount += distr_A2U235[157]*pdf_dZ[-4]/NA*157;
```

```
Mn72.amount += distr_A1U235[73]*pdf_dZ[-3]/NA*73;
Mn73.amount += distr_A1U235[74]*pdf_dZ[-4]/NA*74;
Mn74.amount += distr_A1U235[75]*pdf_dZ[-4]/NA*75;
Mn75.amount += distr_A1U235[76]*pdf_dZ[-4]/NA*76;
Mo100.amount += distr_A1U235[101]*pdf_dZ[3]/NA*101;
Mo101.amount += distr_A1U235[102]*pdf_dZ[2]/NA*102;
Mo102.amount += distr_A1U235[103]*pdf_dZ[2]/NA*103;
Mo103.amount += distr_A1U235[104]*pdf_dZ[2]/NA*104;
Mo104.amount += distr_A1U235[105]*pdf_dZ[1]/NA*105;
Mo105.amount += distr_A1U235[106]*pdf_dZ[1]/NA*106;
Mo106.amount += distr_A1U235[107]*pdf_dZ[0]/NA*107;
Mo107.amount += distr_A1U235[108]*pdf_dZ[0]/NA*108;
Mo108.amount += distr_A1U235[109]*pdf_dZ[0]/NA*109;
Mo109.amount += distr_A1U235[110]*pdf_dZ[-1]/NA*110;
Mo110.amount += distr_A1U235[111]*pdf_dZ[-1]/NA*111;
Mo111.amount += distr_A1U235[112]*pdf_dZ[-2]/NA*112;
Mo112.amount += distr_A1U235[113]*pdf_dZ[-2]/NA*113;
Mo113.amount += distr_A1U235[114]*pdf_dZ[-2]/NA*114;
Mo114.amount += distr_A1U235[115]*pdf_dZ[-3]/NA*115;
Mo115.amount += distr_A1U235[116]*pdf_dZ[-3]/NA*116;
Mo116.amount += distr_A1U235[117]*pdf_dZ[-4]/NA*117;
Mo117.amount += distr_A2U235[118]*pdf_dZ[-4]/NA*118;
Mo118.amount += distr_A2U235[119]*pdf_dZ[-4]/NA*119;
Mo96.amount += distr_A1U235[97]*pdf_dZ[4]/NA*97;
Mo97.amount += distr_A1U235[98]*pdf_dZ[4]/NA*98;
Mo98.amount += distr_A1U235[99]*pdf_dZ[4]/NA*99;
Mo99.amount += distr_A1U235[100]*pdf_dZ[3]/NA*100;
Nb100.amount += distr_A1U235[101]*pdf_dZ[2]/NA*101;
Nb101.amount += distr_A1U235[102]*pdf_dZ[1]/NA*102;
Nb102.amount += distr_A1U235[103]*pdf_dZ[1]/NA*103;
Nb103.amount += distr_A1U235[104]*pdf_dZ[1]/NA*104;
Nb104.amount += distr_A1U235[105]*pdf_dZ[0]/NA*105;
```

```
Nb105.amount += distr_A1U235[106]*pdf_dZ[0]/NA*106;
Nb106.amount += distr_A1U235[107]*pdf_dZ[-1]/NA*107;
Nb107.amount += distr_A1U235[108]*pdf_dZ[-1]/NA*108;
Nb108.amount += distr_A1U235[109]*pdf_dZ[-1]/NA*109;
Nb109.amount += distr_A1U235[110]*pdf_dZ[-2]/NA*110;
Nb110.amount += distr_A1U235[111]*pdf_dZ[-2]/NA*111;
Nb111.amount += distr_A1U235[112]*pdf_dZ[-3]/NA*112;
Nb112.amount += distr_A1U235[113]*pdf_dZ[-3]/NA*113;
Nb113.amount += distr_A1U235[114]*pdf_dZ[-3]/NA*114;
Nb114.amount += distr_A1U235[115]*pdf_dZ[-4]/NA*115;
Nb115.amount += distr_A1U235[116]*pdf_dZ[-4]/NA*116;
Nb93.amount += distr_A1U235[94]*pdf_dZ[4]/NA*94;
Nb94.amount += distr_A1U235[95]*pdf_dZ[4]/NA*95;
Nb95.amount += distr_A1U235[96]*pdf_dZ[4]/NA*96;
Nb96.amount += distr_A1U235[97]*pdf_dZ[3]/NA*97;
Nb97.amount += distr_A1U235[98]*pdf_dZ[3]/NA*98;
Nb98.amount += distr_A1U235[99]*pdf_dZ[3]/NA*99;
Nb99.amount += distr_A1U235[100]*pdf_dZ[2]/NA*100;
Nd142.amount += distr_A2U235[143]*pdf_dZ[4]/NA*143;
Nd143.amount += distr_A2U235[144]*pdf_dZ[4]/NA*144;
Nd144.amount += distr_A2U235[145]*pdf_dZ[3]/NA*145;
Nd145.amount += distr_A2U235[146]*pdf_dZ[3]/NA*146;
Nd146.amount += distr_A2U235[147]*pdf_dZ[3]/NA*147;
Nd147.amount += distr_A2U235[148]*pdf_dZ[2]/NA*148;
Nd148.amount += distr_A2U235[149]*pdf_dZ[2]/NA*149;
Nd149.amount += distr_A2U235[150]*pdf_dZ[1]/NA*150;
Nd150.amount += distr_A2U235[151]*pdf_dZ[1]/NA*151;
Nd151.amount += distr_A2U235[152]*pdf_dZ[1]/NA*152;
Nd152.amount += distr_A2U235[153]*pdf_dZ[0]/NA*153;
Nd153.amount += distr_A2U235[154]*pdf_dZ[0]/NA*154;
Nd154.amount += distr_A2U235[155]*pdf_dZ[-1]/NA*155;
Nd155.amount += distr_A2U235[156]*pdf_dZ[-1]/NA*156;
```

```
Nd156.amount += distr_A2U235[157]*pdf_dZ[-1]/NA*157;
Nd157.amount += distr_A2U235[158]*pdf_dZ[-2]/NA*158;
Nd158.amount += distr_A2U235[159]*pdf_dZ[-2]/NA*159;
Nd159.amount += distr_A2U235[160]*pdf_dZ[-3]/NA*160;
Nd160.amount += distr_A2U235[161]*pdf_dZ[-3]/NA*161;
Nd161.amount += distr_A2U235[162]*pdf_dZ[-3]/NA*162;
Ni72.amount += distr_A1U235[73]*pdf_dZ[0]/NA*73;
Ni73.amount += distr_A1U235[74]*pdf_dZ[-1]/NA*74;
Ni74.amount += distr_A1U235[75]*pdf_dZ[-1]/NA*75;
Ni75.amount += distr_A1U235[76]*pdf_dZ[-1]/NA*76;
Ni76.amount += distr_A1U235[77]*pdf_dZ[-2]/NA*77;
Ni77.amount += distr_A1U235[78]*pdf_dZ[-2]/NA*78;
Ni78.amount += distr_A1U235[79]*pdf_dZ[-3]/NA*79;
Ni79.amount += distr_A1U235[80]*pdf_dZ[-3]/NA*80;
Ni80.amount += distr_A1U235[81]*pdf_dZ[-3]/NA*81;
Ni81.amount += distr_A1U235[82]*pdf_dZ[-4]/NA*82;
Ni82.amount += distr_A1U235[83]*pdf_dZ[-4]/NA*83;
Pd106.amount += distr_A1U235[107]*pdf_dZ[4]/NA*107;
Pd107.amount += distr_A1U235[108]*pdf_dZ[4]/NA*108;
Pd108.amount += distr_A1U235[109]*pdf_dZ[4]/NA*109;
Pd109.amount += distr_A1U235[110]*pdf_dZ[3]/NA*110;
Pd110.amount += distr_A1U235[111]*pdf_dZ[3]/NA*111;
Pd111.amount += distr_A1U235[112]*pdf_dZ[2]/NA*112;
Pd112.amount += distr_A1U235[113]*pdf_dZ[2]/NA*113;
Pd113.amount += distr_A1U235[114]*pdf_dZ[2]/NA*114;
Pd114.amount += distr_A1U235[115]*pdf_dZ[1]/NA*115;
Pd115.amount += distr_A1U235[116]*pdf_dZ[1]/NA*116;
Pd116.amount += distr_A1U235[117]*pdf_dZ[0]/NA*117;
Pd117.amount += distr_A2U235[118]*pdf_dZ[0]/NA*118;
Pd118.amount += distr_A2U235[119]*pdf_dZ[0]/NA*119;
Pd119.amount += distr_A2U235[120]*pdf_dZ[-1]/NA*120;
Pd120.amount += distr_A2U235[121]*pdf_dZ[-1]/NA*121;
```

```
Pd121.amount += distr_A2U235[122]*pdf_dZ[-2]/NA*122;
Pd122.amount += distr_A2U235[123]*pdf_dZ[-2]/NA*123;
Pd123.amount += distr_A2U235[124]*pdf_dZ[-2]/NA*124;
Pd124.amount += distr_A2U235[125]*pdf_dZ[-3]/NA*125;
Pd125.amount += distr_A2U235[126]*pdf_dZ[-3]/NA*126;
Pd126.amount += distr_A2U235[127]*pdf_dZ[-4]/NA*127;
Pd127.amount += distr_A2U235[128]*pdf_dZ[-4]/NA*128;
Pd128.amount += distr_A2U235[129]*pdf_dZ[-4]/NA*129;
Pm144.amount += distr_A2U235[145]*pdf_dZ[4]/NA*145;
Pm145.amount += distr_A2U235[146]*pdf_dZ[4]/NA*146;
Pm146.amount += distr_A2U235[147]*pdf_dZ[4]/NA*147;
Pm147.amount += distr_A2U235[148]*pdf_dZ[3]/NA*148;
Pm148.amount += distr_A2U235[149]*pdf_dZ[3]/NA*149;
Pm149.amount += distr_A2U235[150]*pdf_dZ[2]/NA*150;
Pm150.amount += distr_A2U235[151]*pdf_dZ[2]/NA*151;
Pm151.amount += distr_A2U235[152]*pdf_dZ[2]/NA*152;
Pm152.amount += distr_A2U235[153]*pdf_dZ[1]/NA*153;
Pm153.amount += distr_A2U235[154]*pdf_dZ[1]/NA*154;
Pm154.amount += distr_A2U235[155]*pdf_dZ[0]/NA*155;
Pm155.amount += distr_A2U235[156]*pdf_dZ[0]/NA*156;
Pm156.amount += distr_A2U235[157]*pdf_dZ[0]/NA*157;
Pm157.amount += distr_A2U235[158]*pdf_dZ[-1]/NA*158;
Pm158.amount += distr_A2U235[159]*pdf_dZ[-1]/NA*159;
Pm159.amount += distr_A2U235[160]*pdf_dZ[-2]/NA*160;
Pm160.amount += distr_A2U235[161]*pdf_dZ[-2]/NA*161;
Pm161.amount += distr_A2U235[162]*pdf_dZ[-2]/NA*162;
Pr139.amount += distr_A2U235[140]*pdf_dZ[4]/NA*140;
Pr140.amount += distr_A2U235[141]*pdf_dZ[4]/NA*141;
Pr141.amount += distr_A2U235[142]*pdf_dZ[4]/NA*142;
Pr142.amount += distr_A2U235[143]*pdf_dZ[3]/NA*143;
Pr143.amount += distr_A2U235[144]*pdf_dZ[3]/NA*144;
Pr144.amount += distr_A2U235[145]*pdf_dZ[2]/NA*145;
```

```
Pr145.amount += distr_A2U235[146]*pdf_dZ[2]/NA*146;
Pr146.amount += distr_A2U235[147]*pdf_dZ[2]/NA*147;
Pr147.amount += distr_A2U235[148]*pdf_dZ[1]/NA*148;
Pr148.amount += distr_A2U235[149]*pdf_dZ[1]/NA*149;
Pr149.amount += distr_A2U235[150]*pdf_dZ[0]/NA*150;
Pr150.amount += distr_A2U235[151]*pdf_dZ[0]/NA*151;
Pr151.amount += distr_A2U235[152]*pdf_dZ[0]/NA*152;
Pr152.amount += distr_A2U235[153]*pdf_dZ[-1]/NA*153;
Pr153.amount += distr_A2U235[154]*pdf_dZ[-1]/NA*154;
Pr154.amount += distr_A2U235[155]*pdf_dZ[-2]/NA*155;
Pr155.amount += distr_A2U235[156]*pdf_dZ[-2]/NA*156;
Pr156.amount += distr_A2U235[157]*pdf_dZ[-2]/NA*157;
Pr157.amount += distr_A2U235[158]*pdf_dZ[-3]/NA*158;
Pr158.amount += distr_A2U235[159]*pdf_dZ[-3]/NA*159;
Pr159.amount += distr_A2U235[160]*pdf_dZ[-4]/NA*160;
Pr160.amount += distr_A2U235[161]*pdf_dZ[-4]/NA*161;
Pr161.amount += distr_A2U235[162]*pdf_dZ[-4]/NA*162;
Rb100.amount += distr_A1U235[101]*pdf_dZ[-2]/NA*101;
Rb101.amount += distr_A1U235[102]*pdf_dZ[-3]/NA*102;
Rb102.amount += distr_A1U235[103]*pdf_dZ[-3]/NA*103;
Rb103.amount += distr_A1U235[104]*pdf_dZ[-3]/NA*104;
Rb104.amount += distr_A1U235[105]*pdf_dZ[-4]/NA*105;
Rb105.amount += distr_A1U235[106]*pdf_dZ[-4]/NA*106;
Rb83.amount += distr_A1U235[84]*pdf_dZ[4]/NA*84;
Rb84.amount += distr_A1U235[85]*pdf_dZ[4]/NA*85;
Rb85.amount += distr_A1U235[86]*pdf_dZ[4]/NA*86;
Rb86.amount += distr_A1U235[87]*pdf_dZ[3]/NA*87;
Rb87.amount += distr_A1U235[88]*pdf_dZ[3]/NA*88;
Rb88.amount += distr_A1U235[89]*pdf_dZ[2]/NA*89;
Rb89.amount += distr_A1U235[90]*pdf_dZ[2]/NA*90;
Rb90.amount += distr_A1U235[91]*pdf_dZ[2]/NA*91;
Rb91.amount += distr_A1U235[92]*pdf_dZ[1]/NA*92;
```

```
Rb92.amount += distr_A1U235[93]*pdf_dZ[1]/NA*93;
Rb93.amount += distr_A1U235[94]*pdf_dZ[0]/NA*94;
Rb94.amount += distr_A1U235[95]*pdf_dZ[0]/NA*95;
Rb95.amount += distr_A1U235[96]*pdf_dZ[0]/NA*96;
Rb96.amount += distr_A1U235[97]*pdf_dZ[-1]/NA*97;
Rb97.amount += distr_A1U235[98]*pdf_dZ[-1]/NA*98;
Rb98.amount += distr_A1U235[99]*pdf_dZ[-1]/NA*99;
Rb99.amount += distr_A1U235[100]*pdf_dZ[-2]/NA*100;
Rh104.amount += distr_A1U235[105]*pdf_dZ[4]/NA*105;
Rh105.amount += distr_A1U235[106]*pdf_dZ[4]/NA*106;
Rh106.amount += distr_A1U235[107]*pdf_dZ[3]/NA*107;
Rh107.amount += distr_A1U235[108]*pdf_dZ[3]/NA*108;
Rh108.amount += distr_A1U235[109]*pdf_dZ[3]/NA*109;
Rh109.amount += distr_A1U235[110]*pdf_dZ[2]/NA*110;
Rh110.amount += distr_A1U235[111]*pdf_dZ[2]/NA*111;
Rh111.amount += distr_A1U235[112]*pdf_dZ[1]/NA*112;
Rh112.amount += distr_A1U235[113]*pdf_dZ[1]/NA*113;
Rh113.amount += distr_A1U235[114]*pdf_dZ[1]/NA*114;
Rh114.amount += distr_A1U235[115]*pdf_dZ[0]/NA*115;
Rh115.amount += distr_A1U235[116]*pdf_dZ[0]/NA*116;
Rh116.amount += distr_A1U235[117]*pdf_dZ[-1]/NA*117;
Rh117.amount += distr_A2U235[118]*pdf_dZ[-1]/NA*118;
Rh118.amount += distr_A2U235[119]*pdf_dZ[-1]/NA*119;
Rh119.amount += distr_A2U235[120]*pdf_dZ[-2]/NA*120;
Rh120.amount += distr_A2U235[121]*pdf_dZ[-2]/NA*121;
Rh121.amount += distr_A2U235[122]*pdf_dZ[-3]/NA*122;
Rh122.amount += distr_A2U235[123]*pdf_dZ[-3]/NA*123;
Rh123.amount += distr_A2U235[124]*pdf_dZ[-3]/NA*124;
Rh124.amount += distr_A2U235[125]*pdf_dZ[-4]/NA*125;
Rh125.amount += distr_A2U235[126]*pdf_dZ[-4]/NA*126;
Ru101.amount += distr_A1U235[102]*pdf_dZ[4]/NA*102;
Ru102.amount += distr_A1U235[103]*pdf_dZ[4]/NA*103;
```

```
Ru103.amount += distr_A1U235[104]*pdf_dZ[4]/NA*104;
Ru104.amount += distr_A1U235[105]*pdf_dZ[3]/NA*105;
Ru105.amount += distr_A1U235[106]*pdf_dZ[3]/NA*106;
Ru106.amount += distr_A1U235[107]*pdf_dZ[2]/NA*107;
Ru107.amount += distr_A1U235[108]*pdf_dZ[2]/NA*108;
Ru108.amount += distr_A1U235[109]*pdf_dZ[2]/NA*109;
Ru109.amount += distr_A1U235[110]*pdf_dZ[1]/NA*110;
Ru110.amount += distr_A1U235[111]*pdf_dZ[1]/NA*111;
Ru111.amount += distr_A1U235[112]*pdf_dZ[0]/NA*112;
Ru112.amount += distr_A1U235[113]*pdf_dZ[0]/NA*113;
Ru113.amount += distr_A1U235[114]*pdf_dZ[0]/NA*114;
Ru114.amount += distr_A1U235[115]*pdf_dZ[-1]/NA*115;
Ru115.amount += distr_A1U235[116]*pdf_dZ[-1]/NA*116;
Ru116.amount += distr_A1U235[117]*pdf_dZ[-2]/NA*117;
Ru117.amount += distr_A2U235[118]*pdf_dZ[-2]/NA*118;
Ru118.amount += distr_A2U235[119]*pdf_dZ[-2]/NA*119;
Ru119.amount += distr_A2U235[120]*pdf_dZ[-3]/NA*120;
Ru120.amount += distr_A2U235[121]*pdf_dZ[-3]/NA*121;
Ru121.amount += distr_A2U235[122]*pdf_dZ[-4]/NA*122;
Ru122.amount += distr_A2U235[123]*pdf_dZ[-4]/NA*123;
Ru123.amount += distr_A2U235[124]*pdf_dZ[-4]/NA*124;
Sb119.amount += distr_A2U235[120]*pdf_dZ[4]/NA*120;
Sb120.amount += distr_A2U235[121]*pdf_dZ[4]/NA*121;
Sb121.amount += distr_A2U235[122]*pdf_dZ[3]/NA*122;
Sb122.amount += distr_A2U235[123]*pdf_dZ[3]/NA*123;
Sb123.amount += distr_A2U235[124]*pdf_dZ[3]/NA*124;
Sb124.amount += distr_A2U235[125]*pdf_dZ[2]/NA*125;
Sb125.amount += distr_A2U235[126]*pdf_dZ[2]/NA*126;
Sb126.amount += distr_A2U235[127]*pdf_dZ[1]/NA*127;
Sb127.amount += distr_A2U235[128]*pdf_dZ[1]/NA*128;
Sb128.amount += distr_A2U235[129]*pdf_dZ[1]/NA*129;
Sb129.amount += distr_A2U235[130]*pdf_dZ[0]/NA*130;
```

```
Sb130.amount += distr_A2U235[131]*pdf_dZ[0]/NA*131;
Sb131.amount += distr_A2U235[132]*pdf_dZ[0]/NA*132;
Sb132.amount += distr_A2U235[133]*pdf_dZ[-1]/NA*133;
Sb133.amount += distr_A2U235[134]*pdf_dZ[-1]/NA*134;
Sb134.amount += distr_A2U235[135]*pdf_dZ[-2]/NA*135;
Sb135.amount += distr_A2U235[136]*pdf_dZ[-2]/NA*136;
Sb136.amount += distr_A2U235[137]*pdf_dZ[-2]/NA*137;
Sb137.amount += distr_A2U235[138]*pdf_dZ[-3]/NA*138;
Sb138.amount += distr_A2U235[139]*pdf_dZ[-3]/NA*139;
Sb139.amount += distr_A2U235[140]*pdf_dZ[-4]/NA*140;
Sb140.amount += distr_A2U235[141]*pdf_dZ[-4]/NA*141;
Sb141.amount += distr_A2U235[142]*pdf_dZ[-4]/NA*142;
Se76.amount += distr_A1U235[77]*pdf_dZ[4]/NA*77;
Se77.amount += distr_A1U235[78]*pdf_dZ[4]/NA*78;
Se78.amount += distr_A1U235[79]*pdf_dZ[3]/NA*79;
Se79.amount += distr_A1U235[80]*pdf_dZ[3]/NA*80;
Se80.amount += distr_A1U235[81]*pdf_dZ[3]/NA*81;
Se81.amount += distr_A1U235[82]*pdf_dZ[2]/NA*82;
Se82.amount += distr_A1U235[83]*pdf_dZ[2]/NA*83;
Se83.amount += distr_A1U235[84]*pdf_dZ[1]/NA*84;
Se84.amount += distr_A1U235[85]*pdf_dZ[1]/NA*85;
Se85.amount += distr_A1U235[86]*pdf_dZ[1]/NA*86;
Se86.amount += distr_A1U235[87]*pdf_dZ[0]/NA*87;
Se87.amount += distr_A1U235[88]*pdf_dZ[0]/NA*88;
Se88.amount += distr_A1U235[89]*pdf_dZ[-1]/NA*89;
Se89.amount += distr_A1U235[90]*pdf_dZ[-1]/NA*90;
Se90.amount += distr_A1U235[91]*pdf_dZ[-1]/NA*91;
Se91.amount += distr_A1U235[92]*pdf_dZ[-2]/NA*92;
Se92.amount += distr_A1U235[93]*pdf_dZ[-2]/NA*93;
Se93.amount += distr_A1U235[94]*pdf_dZ[-3]/NA*94;
Se94.amount += distr_A1U235[95]*pdf_dZ[-3]/NA*95;
Se95.amount += distr_A1U235[96]*pdf_dZ[-3]/NA*96;
```

```
Se96.amount += distr_A1U235[97]*pdf_dZ[-4]/NA*97;
Se97.amount += distr_A1U235[98]*pdf_dZ[-4]/NA*98;
Se98.amount += distr_A1U235[99]*pdf_dZ[-4]/NA*99;
Sm147.amount += distr_A2U235[148]*pdf_dZ[4]/NA*148;
Sm148.amount += distr_A2U235[149]*pdf_dZ[4]/NA*149;
Sm149.amount += distr_A2U235[150]*pdf_dZ[3]/NA*150;
Sm150.amount += distr_A2U235[151]*pdf_dZ[3]/NA*151;
Sm151.amount += distr_A2U235[152]*pdf_dZ[3]/NA*152;
Sm152.amount += distr_A2U235[153]*pdf_dZ[2]/NA*153;
Sm153.amount += distr_A2U235[154]*pdf_dZ[2]/NA*154;
Sm154.amount += distr_A2U235[155]*pdf_dZ[1]/NA*155;
Sm155.amount += distr_A2U235[156]*pdf_dZ[1]/NA*156;
Sm156.amount += distr_A2U235[157]*pdf_dZ[1]/NA*157;
Sm157.amount += distr_A2U235[158]*pdf_dZ[0]/NA*158;
Sm158.amount += distr_A2U235[159]*pdf_dZ[0]/NA*159;
Sm159.amount += distr_A2U235[160]*pdf_dZ[-1]/NA*160;
Sm160.amount += distr_A2U235[161]*pdf_dZ[-1]/NA*161;
Sm161.amount += distr_A2U235[162]*pdf_dZ[-1]/NA*162;
Sn116.amount += distr_A1U235[117]*pdf_dZ[4]/NA*117;
Sn117.amount += distr_A2U235[118]*pdf_dZ[4]/NA*118;
Sn118.amount += distr_A2U235[119]*pdf_dZ[4]/NA*119;
Sn119.amount += distr_A2U235[120]*pdf_dZ[3]/NA*120;
Sn120.amount += distr_A2U235[121]*pdf_dZ[3]/NA*121;
Sn121.amount += distr_A2U235[122]*pdf_dZ[2]/NA*122;
Sn122.amount += distr_A2U235[123]*pdf_dZ[2]/NA*123;
Sn123.amount += distr_A2U235[124]*pdf_dZ[2]/NA*124;
Sn124.amount += distr_A2U235[125]*pdf_dZ[1]/NA*125;
Sn125.amount += distr_A2U235[126]*pdf_dZ[1]/NA*126;
Sn126.amount += distr_A2U235[127]*pdf_dZ[0]/NA*127;
Sn127.amount += distr_A2U235[128]*pdf_dZ[0]/NA*128;
Sn128.amount += distr_A2U235[129]*pdf_dZ[0]/NA*129;
Sn129.amount += distr_A2U235[130]*pdf_dZ[-1]/NA*130;
```

```
Sn130.amount += distr_A2U235[131]*pdf_dZ[-1]/NA*131;
Sn131.amount += distr_A2U235[132]*pdf_dZ[-1]/NA*132;
Sn132.amount += distr_A2U235[133]*pdf_dZ[-2]/NA*133;
Sn133.amount += distr_A2U235[134]*pdf_dZ[-2]/NA*134;
Sn134.amount += distr_A2U235[135]*pdf_dZ[-3]/NA*135;
Sn135.amount += distr_A2U235[136]*pdf_dZ[-3]/NA*136;
Sn136.amount += distr_A2U235[137]*pdf_dZ[-3]/NA*137;
Sn137.amount += distr_A2U235[138]*pdf_dZ[-4]/NA*138;
Sn138.amount += distr_A2U235[139]*pdf_dZ[-4]/NA*139;
Sr100.amount += distr_A1U235[101]*pdf_dZ[-1]/NA*101;
Sr101.amount += distr_A1U235[102]*pdf_dZ[-2]/NA*102;
Sr102.amount += distr_A1U235[103]*pdf_dZ[-2]/NA*103;
Sr103.amount += distr_A1U235[104]*pdf_dZ[-2]/NA*104;
Sr104.amount += distr_A1U235[105]*pdf_dZ[-3]/NA*105;
Sr105.amount += distr_A1U235[106]*pdf_dZ[-3]/NA*106;
Sr106.amount += distr_A1U235[107]*pdf_dZ[-4]/NA*107;
Sr107.amount += distr_A1U235[108]*pdf_dZ[-4]/NA*108;
Sr108.amount += distr_A1U235[109]*pdf_dZ[-4]/NA*109;
Sr86.amount += distr_A1U235[87]*pdf_dZ[4]/NA*87;
Sr87.amount += distr_A1U235[88]*pdf_dZ[4]/NA*88;
Sr88.amount += distr_A1U235[89]*pdf_dZ[3]/NA*89;
Sr89.amount += distr_A1U235[90]*pdf_dZ[3]/NA*90;
Sr90.amount += distr_A1U235[91]*pdf_dZ[3]/NA*91;
Sr91.amount += distr_A1U235[92]*pdf_dZ[2]/NA*92;
Sr92.amount += distr_A1U235[93]*pdf_dZ[2]/NA*93;
Sr93.amount += distr_A1U235[94]*pdf_dZ[1]/NA*94;
Sr94.amount += distr_A1U235[95]*pdf_dZ[1]/NA*95;
Sr95.amount += distr_A1U235[96]*pdf_dZ[1]/NA*96;
Sr96.amount += distr_A1U235[97]*pdf_dZ[0]/NA*97;
Sr97.amount += distr_A1U235[98]*pdf_dZ[0]/NA*98;
Sr98.amount += distr_A1U235[99]*pdf_dZ[0]/NA*99;
Sr99.amount += distr_A1U235[100]*pdf_dZ[-1]/NA*100;
```

```
Tb154.amount += distr_A2U235[155]*pdf_dZ[4]/NA*155;
Tb155.amount += distr_A2U235[156]*pdf_dZ[4]/NA*156;
Tb156.amount += distr_A2U235[157]*pdf_dZ[4]/NA*157;
Tb157.amount += distr_A2U235[158]*pdf_dZ[3]/NA*158;
Tb158.amount += distr_A2U235[159]*pdf_dZ[3]/NA*159;
Tb159.amount += distr_A2U235[160]*pdf_dZ[2]/NA*160;
Tb160.amount += distr_A2U235[161]*pdf_dZ[2]/NA*161;
Tb161.amount += distr_A2U235[162]*pdf_dZ[2]/NA*162;
Tc100.amount += distr_A1U235[101]*pdf_dZ[4]/NA*101;
Tc101.amount += distr_A1U235[102]*pdf_dZ[3]/NA*102;
Tc102.amount += distr_A1U235[103]*pdf_dZ[3]/NA*103;
Tc103.amount += distr_A1U235[104]*pdf_dZ[3]/NA*104;
Tc104.amount += distr_A1U235[105]*pdf_dZ[2]/NA*105;
Tc105.amount += distr_A1U235[106]*pdf_dZ[2]/NA*106;
Tc106.amount += distr_A1U235[107]*pdf_dZ[1]/NA*107;
Tc107.amount += distr_A1U235[108]*pdf_dZ[1]/NA*108;
Tc108.amount += distr_A1U235[109]*pdf_dZ[1]/NA*109;
Tc109.amount += distr_A1U235[110]*pdf_dZ[0]/NA*110;
Tc110.amount += distr_A1U235[111]*pdf_dZ[0]/NA*111;
Tc111.amount += distr_A1U235[112]*pdf_dZ[-1]/NA*112;
Tc112.amount += distr_A1U235[113]*pdf_dZ[-1]/NA*113;
Tc113.amount += distr_A1U235[114]*pdf_dZ[-1]/NA*114;
Tc114.amount += distr_A1U235[115]*pdf_dZ[-2]/NA*115;
Tc115.amount += distr_A1U235[116]*pdf_dZ[-2]/NA*116;
Tc116.amount += distr_A1U235[117]*pdf_dZ[-3]/NA*117;
Tc117.amount += distr_A2U235[118]*pdf_dZ[-3]/NA*118;
Tc118.amount += distr_A2U235[119]*pdf_dZ[-3]/NA*119;
Tc119.amount += distr_A2U235[120]*pdf_dZ[-4]/NA*120;
Tc120.amount += distr_A2U235[121]*pdf_dZ[-4]/NA*121;
Tc99.amount += distr_A1U235[100]*pdf_dZ[4]/NA*100;
Te121.amount += distr_A2U235[122]*pdf_dZ[4]/NA*122;
Te122.amount += distr_A2U235[123]*pdf_dZ[4]/NA*123;
```

```
Te123.amount += distr_A2U235[124]*pdf_dZ[4]/NA*124;
Te124.amount += distr_A2U235[125]*pdf_dZ[3]/NA*125;
Te125.amount += distr_A2U235[126]*pdf_dZ[3]/NA*126;
Te126.amount += distr_A2U235[127]*pdf_dZ[2]/NA*127;
Te127.amount += distr_A2U235[128]*pdf_dZ[2]/NA*128;
Te128.amount += distr_A2U235[129]*pdf_dZ[2]/NA*129;
Te129.amount += distr_A2U235[130]*pdf_dZ[1]/NA*130;
Te130.amount += distr_A2U235[131]*pdf_dZ[1]/NA*131;
Te131.amount += distr_A2U235[132]*pdf_dZ[1]/NA*132;
Te132.amount += distr_A2U235[133]*pdf_dZ[0]/NA*133;
Te133.amount += distr_A2U235[134]*pdf_dZ[0]/NA*134;
Te134.amount += distr_A2U235[135]*pdf_dZ[-1]/NA*135;
Te135.amount += distr_A2U235[136]*pdf_dZ[-1]/NA*136;
Te136.amount += distr_A2U235[137]*pdf_dZ[-1]/NA*137;
Te137.amount += distr_A2U235[138]*pdf_dZ[-2]/NA*138;
Te138.amount += distr_A2U235[139]*pdf_dZ[-2]/NA*139;
Te139.amount += distr_A2U235[140]*pdf_dZ[-3]/NA*140;
Te140.amount += distr_A2U235[141]*pdf_dZ[-3]/NA*141;
Te141.amount += distr_A2U235[142]*pdf_dZ[-3]/NA*142;
Te142.amount += distr_A2U235[143]*pdf_dZ[-4]/NA*143;
Te143.amount += distr_A2U235[144]*pdf_dZ[-4]/NA*144;
Xe126.amount += distr_A2U235[127]*pdf_dZ[4]/NA*127;
Xe127.amount += distr_A2U235[128]*pdf_dZ[4]/NA*128;
Xe128.amount += distr_A2U235[129]*pdf_dZ[4]/NA*129;
Xe129.amount += distr_A2U235[130]*pdf_dZ[3]/NA*130;
Xe130.amount += distr_A2U235[131]*pdf_dZ[3]/NA*131;
Xe131.amount += distr_A2U235[132]*pdf_dZ[3]/NA*132;
Xe132.amount += distr_A2U235[133]*pdf_dZ[2]/NA*133;
Xe133.amount += distr_A2U235[134]*pdf_dZ[2]/NA*134;
Xe134.amount += distr_A2U235[135]*pdf_dZ[1]/NA*135;
Xe135.amount += distr_A2U235[136]*pdf_dZ[1]/NA*136;
Xe136.amount += distr_A2U235[137]*pdf_dZ[1]/NA*137;
```

```
Xe137.amount += distr_A2U235[138]*pdf_dZ[0]/NA*138;
Xe138.amount += distr_A2U235[139]*pdf_dZ[0]/NA*139;
Xe139.amount += distr_A2U235[140]*pdf_dZ[-1]/NA*140;
Xe140.amount += distr_A2U235[141]*pdf_dZ[-1]/NA*141;
Xe141.amount += distr_A2U235[142]*pdf_dZ[-1]/NA*142;
Xe142.amount += distr_A2U235[143]*pdf_dZ[-2]/NA*143;
Xe143.amount += distr_A2U235[144]*pdf_dZ[-2]/NA*144;
Xe144.amount += distr_A2U235[145]*pdf_dZ[-3]/NA*145;
Xe145.amount += distr_A2U235[146]*pdf_dZ[-3]/NA*146;
Xe146.amount += distr_A2U235[147]*pdf_dZ[-3]/NA*147;
Xe147.amount += distr_A2U235[148]*pdf_dZ[-4]/NA*148;
Xe148.amount += distr_A2U235[149]*pdf_dZ[-4]/NA*149;
Y100.amount += distr_A1U235[101]*pdf_dZ[0]/NA*101;
Y101.amount += distr_A1U235[102]*pdf_dZ[-1]/NA*102;
Y102.amount += distr_A1U235[103]*pdf_dZ[-1]/NA*103;
Y103.amount += distr_A1U235[104]*pdf_dZ[-1]/NA*104;
Y104.amount += distr_A1U235[105]*pdf_dZ[-2]/NA*105;
Y105.amount += distr_A1U235[106]*pdf_dZ[-2]/NA*106;
Y106.amount += distr_A1U235[107]*pdf_dZ[-3]/NA*107;
Y107.amount += distr_A1U235[108]*pdf_dZ[-3]/NA*108;
Y108.amount += distr_A1U235[109]*pdf_dZ[-3]/NA*109;
Y109.amount += distr_A1U235[110]*pdf_dZ[-4]/NA*110;
Y110.amount += distr_A1U235[111]*pdf_dZ[-4]/NA*111;
Y88.amount += distr_A1U235[89]*pdf_dZ[4]/NA*89;
Y89.amount += distr_A1U235[90]*pdf_dZ[4]/NA*90;
Y90.amount += distr_A1U235[91]*pdf_dZ[4]/NA*91;
Y91.amount += distr_A1U235[92]*pdf_dZ[3]/NA*92;
Y92.amount += distr_A1U235[93]*pdf_dZ[3]/NA*93;
Y93.amount += distr_A1U235[94]*pdf_dZ[2]/NA*94;
Y94.amount += distr_A1U235[95]*pdf_dZ[2]/NA*95;
Y95.amount += distr_A1U235[96]*pdf_dZ[2]/NA*96;
Y96.amount += distr_A1U235[97]*pdf_dZ[1]/NA*97;
```

```
Y97.amount += distr_A1U235[98]*pdf_dZ[1]/NA*98;
Y98.amount += distr_A1U235[99]*pdf_dZ[1]/NA*99;
Y99.amount += distr_A1U235[100]*pdf_dZ[0]/NA*100;
Zn72.amount += distr_A1U235[73]*pdf_dZ[2]/NA*73;
Zn73.amount += distr_A1U235[74]*pdf_dZ[1]/NA*74;
Zn74.amount += distr_A1U235[75]*pdf_dZ[1]/NA*75;
Zn75.amount += distr_A1U235[76]*pdf_dZ[1]/NA*76;
Zn76.amount += distr_A1U235[77]*pdf_dZ[0]/NA*77;
Zn77.amount += distr_A1U235[78]*pdf_dZ[0]/NA*78;
Zn78.amount += distr_A1U235[79]*pdf_dZ[-1]/NA*79;
Zn79.amount += distr_A1U235[80]*pdf_dZ[-1]/NA*80;
Zn80.amount += distr_A1U235[81]*pdf_dZ[-1]/NA*81;
Zn81.amount += distr_A1U235[82]*pdf_dZ[-2]/NA*82;
Zn82.amount += distr_A1U235[83]*pdf_dZ[-2]/NA*83;
Zn83.amount += distr_A1U235[84]*pdf_dZ[-3]/NA*84;
Zn84.amount += distr_A1U235[85]*pdf_dZ[-3]/NA*85;
Zn85.amount += distr_A1U235[86]*pdf_dZ[-3]/NA*86;
Zn86.amount += distr_A1U235[87]*pdf_dZ[-4]/NA*87;
Zn87.amount += distr_A1U235[88]*pdf_dZ[-4]/NA*88;
Zr100.amount += distr_A1U235[101]*pdf_dZ[1]/NA*101;
Zr101.amount += distr_A1U235[102]*pdf_dZ[0]/NA*102;
Zr102.amount += distr_A1U235[103]*pdf_dZ[0]/NA*103;
Zr103.amount += distr_A1U235[104]*pdf_dZ[0]/NA*104;
Zr104.amount += distr_A1U235[105]*pdf_dZ[-1]/NA*105;
Zr105.amount += distr_A1U235[106]*pdf_dZ[-1]/NA*106;
Zr106.amount += distr_A1U235[107]*pdf_dZ[-2]/NA*107;
Zr107.amount += distr_A1U235[108]*pdf_dZ[-2]/NA*108;
Zr108.amount += distr_A1U235[109]*pdf_dZ[-2]/NA*109;
Zr109.amount += distr_A1U235[110]*pdf_dZ[-3]/NA*110;
Zr110.amount += distr_A1U235[111]*pdf_dZ[-3]/NA*111;
Zr111.amount += distr_A1U235[112]*pdf_dZ[-4]/NA*112;
Zr112.amount += distr_A1U235[113]*pdf_dZ[-4]/NA*113;
```

```
Zr113.amount += distr_A1U235[114]*pdf_dZ[-4]/NA*114;
Zr91.amount += distr_A1U235[92]*pdf_dZ[4]/NA*92;
Zr92.amount += distr_A1U235[93]*pdf_dZ[4]/NA*93;
Zr93.amount += distr_A1U235[94]*pdf_dZ[3]/NA*94;
Zr94.amount += distr_A1U235[95]*pdf_dZ[3]/NA*95;
Zr95.amount += distr_A1U235[96]*pdf_dZ[3]/NA*96;
Zr96.amount += distr_A1U235[97]*pdf_dZ[2]/NA*97;
Zr97.amount += distr_A1U235[98]*pdf_dZ[2]/NA*98;
Zr98.amount += distr_A1U235[99]*pdf_dZ[2]/NA*99;
Zr99.amount += distr_A1U235[100]*pdf_dZ[1]/NA*100;
```

Appendix VII

Final results of Experiment 2

This section shows the results of running the simulation after 20 seconds for the Example. The reactor is fueled with natural Uranium which consists of 0.711 % of ^{235}U and 99.3 % of ^{289}U. The average prompt neutron lifetime is assumed to be 20 milliseconds, and this is the cycle time also. The report time interval is 1 second, which means there are 50 cycles in each report period (see Fig. 3.3). The simulation results are reported at the end of each report period.

The number of fissions for each report time period, the fuel consumption and fuel remaining are listed, along with the number of fission neutrons, the number of beta positive particles, the number of beta negative particles, as well as the energy of fission products, the energy of fission neutrons, the energy of gamma particles, and the energy of beta particles. At the end of simulation, the final composition of fission products are shown in this Appendix. These results show a subcritical state for the reactor, where the number of fissions decreases with each report period.

```
Simulation period : 20.000 s
Report interval   : 1.000 s
Cycle time        : 0.020 s

Summary report: 1

Number of fissions                      : 3.345e+20
Amount of fuel fissioned                : 1.305e-01 gram
Fuel left (U235)                        : 7.000e+03 gram
Fuel left (U238)                        : 9.930e+05 gram
Number neutrons released                : 8.128e+20
Neutron energy released                 : 1.603e+21 Mev
Gamma energy released                   : 8.746e+20 Mev
Energy of fission fragment F1           : 2.284e+22 Mev
Energy of fission fragment F2           : 3.301e+22 Mev
Beta Positive particles released        : 1.068e+12
Beta positive energy released           : 4.271e+11 Mev
Beta negative particles released        : 1.107e+21
Beta negative energy released           : 4.428e+20 Mev
Total energy released                   : 5.877e+22 Mev

Energy release per fission              : 1.757e+02 Mev
Energy of fission fragment per fission  : 1.670e+02 Mev
Energy neutron per fission              : 4.793e+00 Mev
Energy gamma per fission                : 2.615e+00 Mev
Energy beta per fission                 : 1.324e+00 Mev
Average number neutrons per fission     : 2.4298


Summary report: 2

Number of fissions                      : 1.023e+18
Amount of fuel fissioned                : 3.992e-04 gram
Fuel left (U235)                        : 7.000e+03 gram
Fuel left (U238)                        : 9.930e+05 gram
Number neutrons released                : 2.486e+18
Neutron energy released                 : 4.903e+18 Mev
Gamma energy released                   : 2.675e+18 Mev
Energy of fission fragment F1           : 6.986e+19 Mev
Energy of fission fragment F2           : 1.010e+20 Mev
Beta Positive particles released        : 2.294e+10
Beta positive energy released           : 9.178e+09 Mev
Beta negative particles released        : 4.499e+20
Beta negative energy released           : 1.800e+20 Mev
Total energy released                   : 3.584e+20 Mev

Energy release per fission              : 3.503e+02 Mev
Energy of fission fragment per fission  : 1.670e+02 Mev
Energy neutron per fission              : 4.793e+00 Mev
Energy gamma per fission                : 2.615e+00 Mev
Energy beta per fission                 : 1.759e+02 Mev
Average number neutrons per fission     : 2.4301
```

Summary report: 3

Number of fissions	: 3.157e+15
Amount of fuel fissioned	: 1.232e-06 gram
Fuel left (U235)	: 7.000e+03 gram
Fuel left (U238)	: 9.930e+05 gram
Number neutrons released	: 7.671e+15
Neutron energy released	: 1.513e+16 Mev
Gamma energy released	: 8.254e+15 Mev
Energy of fission fragment F1	: 2.156e+17 Mev
Energy of fission fragment F2	: 3.116e+17 Mev
Beta Positive particles released	: 1.402e+08
Beta positive energy released	: 5.610e+07 Mev
Beta negative particles released	: 2.274e+20
Beta negative energy released	: 9.095e+19 Mev
Total energy released	: 9.151e+19 Mev
Energy release per fission	: 2.899e+04 Mev
Energy of fission fragment per fission	: 1.670e+02 Mev
Energy neutron per fission	: 4.793e+00 Mev
Energy gamma per fission	: 2.615e+00 Mev
Energy beta per fission	: 2.881e+04 Mev
Average number neutrons per fission	: 2.4298

Summary report: 4

Number of fissions	: 9.752e+12
Amount of fuel fissioned	: 3.805e-09 gram
Fuel left (U235)	: 7.000e+03 gram
Fuel left (U238)	: 9.930e+05 gram
Number neutrons released	: 2.370e+13
Neutron energy released	: 4.674e+13 Mev
Gamma energy released	: 2.550e+13 Mev
Energy of fission fragment F1	: 6.659e+14 Mev
Energy of fission fragment F2	: 9.625e+14 Mev
Beta Positive particles released	: 6.135e+05
Beta positive energy released	: 2.454e+05 Mev
Beta negative particles released	: 1.411e+20
Beta negative energy released	: 5.644e+19 Mev
Total energy released	: 5.644e+19 Mev
Energy release per fission	: 5.787e+06 Mev
Energy of fission fragment per fission	: 1.670e+02 Mev
Energy neutron per fission	: 4.793e+00 Mev
Energy gamma per fission	: 2.615e+00 Mev
Energy beta per fission	: 5.787e+06 Mev
Average number neutrons per fission	: 2.4303

Summary report: 5

Number of fissions	: 3.013e+10
Amount of fuel fissioned	: 1.176e-11 gram

Fuel left (U235)	: 7.000e+03 gram
Fuel left (U238)	: 9.930e+05 gram
Number neutrons released	: 7.322e+10
Neutron energy released	: 1.444e+11 Mev
Gamma energy released	: 7.877e+10 Mev
Energy of fission fragment F1	: 2.057e+12 Mev
Energy of fission fragment F2	: 2.974e+12 Mev
Beta Positive particles released	: 2.282e+03
Beta positive energy released	: 9.129e+02 Mev
Beta negative particles released	: 9.935e+19
Beta negative energy released	: 3.974e+19 Mev
Total energy released	: 3.974e+19 Mev
Energy release per fission	: 1.319e+09 Mev
Energy of fission fragment per fission	: 1.670e+02 Mev
Energy neutron per fission	: 4.793e+00 Mev
Energy gamma per fission	: 2.615e+00 Mev
Energy beta per fission	: 1.319e+09 Mev
Average number neutrons per fission	: 2.4301

Summary report: 6

Number of fissions	: 9.309e+07
Amount of fuel fissioned	: 3.633e-14 gram
Fuel left (U235)	: 7.000e+03 gram
Fuel left (U238)	: 9.930e+05 gram
Number neutrons released	: 2.262e+08
Neutron energy released	: 4.461e+08 Mev
Gamma energy released	: 2.434e+08 Mev
Energy of fission fragment F1	: 6.357e+09 Mev
Energy of fission fragment F2	: 9.188e+09 Mev
Beta Positive particles released	: 7.891e+00
Beta positive energy released	: 3.157e+00 Mev
Beta negative particles released	: 7.549e+19
Beta negative energy released	: 3.020e+19 Mev
Total energy released	: 3.020e+19 Mev
Energy release per fission	: 3.244e+11 Mev
Energy of fission fragment per fission	: 1.670e+02 Mev
Energy neutron per fission	: 4.793e+00 Mev
Energy gamma per fission	: 2.615e+00 Mev
Energy beta per fission	: 3.244e+11 Mev
Average number neutrons per fission	: 2.4229

Summary report: 7

Number of fissions	: 2.876e+05
Amount of fuel fissioned	: 1.122e-16 gram
Fuel left (U235)	: 7.000e+03 gram
Fuel left (U238)	: 9.930e+05 gram
Number neutrons released	: 6.989e+05

Neutron energy released	: 1.378e+06 Mev
Gamma energy released	: 7.520e+05 Mev
Energy of fission fragment F1	: 1.964e+07 Mev
Energy of fission fragment F2	: 2.839e+07 Mev
Beta Positive particles released	: 4.658e-03
Beta positive energy released	: 1.863e-03 Mev
Beta negative particles released	: 6.028e+19
Beta negative energy released	: 2.411e+19 Mev
Total energy released	: 2.411e+19 Mev
Energy release per fission	: 8.384e+13 Mev
Energy of fission fragment per fission	: 1.670e+02 Mev
Energy neutron per fission	: 4.793e+00 Mev
Energy gamma per fission	: 2.615e+00 Mev
Energy beta per fission	: 8.384e+13 Mev
Average number neutrons per fission	: 2.4301

Summary report: 8

Number of fissions	: 8.886e+02
Amount of fuel fissioned	: 3.468e-19 gram
Fuel left (U235)	: 7.000e+03 gram
Fuel left (U238)	: 9.930e+05 gram
Number neutrons released	: 2.159e+03
Neutron energy released	: 4.259e+03 Mev
Gamma energy released	: 2.323e+03 Mev
Energy of fission fragment F1	: 6.068e+04 Mev
Energy of fission fragment F2	: 8.771e+04 Mev
Beta Positive particles released	: 0.000e+00
Beta positive energy released	: 0.000e+00 Mev
Beta negative particles released	: 4.974e+19
Beta negative energy released	: 1.989e+19 Mev
Total energy released	: 1.989e+19 Mev
Energy release per fission	: 2.239e+16 Mev
Energy of fission fragment per fission	: 1.670e+02 Mev
Energy neutron per fission	: 4.793e+00 Mev
Energy gamma per fission	: 2.615e+00 Mev
Energy beta per fission	: 2.239e+16 Mev
Average number neutrons per fission	: 2.4297

Summary report: 9

Number of fissions	: 2.745e+00
Amount of fuel fissioned	: 1.071e-21 gram
Fuel left (U235)	: 7.000e+03 gram
Fuel left (U238)	: 9.930e+05 gram
Number neutrons released	: 6.672e+00
Neutron energy released	: 1.316e+01 Mev
Gamma energy released	: 7.178e+00 Mev
Energy of fission fragment F1	: 1.875e+02 Mev

Energy of fission fragment F2	: 2.710e+02 Mev
Beta Positive particles released	: 0.000e+00
Beta positive energy released	: 0.000e+00 Mev
Beta negative particles released	: 4.197e+19
Beta negative energy released	: 1.679e+19 Mev
Total energy released	: 1.679e+19 Mev
Energy release per fission	: 6.115e+18 Mev
Energy of fission fragment per fission	: 1.670e+02 Mev
Energy neutron per fission	: 4.793e+00 Mev
Energy gamma per fission	: 2.615e+00 Mev
Energy beta per fission	: 6.115e+18 Mev
Average number neutrons per fission	: 2.4306

Summary report: 10

Fuel left (U235)	: 7.000e+03 gram
Fuel left (U238)	: 9.930e+05 gram
Gamma energy released	: 2.218e-02 Mev
Beta Positive particles released	: 0.000e+00
Beta positive energy released	: 0.000e+00 Mev
Beta negative particles released	: 3.598e+19
Beta negative energy released	: 1.439e+19 Mev
Total energy released	: 1.439e+19 Mev

Summary report: 11

Fuel left (U235)	: 7.000e+03 gram
Fuel left (U238)	: 9.930e+05 gram
Gamma energy released	: 6.852e-05 Mev
Beta Positive particles released	: 0.000e+00
Beta positive energy released	: 0.000e+00 Mev
Beta negative particles released	: 3.120e+19
Beta negative energy released	: 1.248e+19 Mev
Total energy released	: 1.248e+19 Mev

Summary report: 12

Fuel left (U235)	: 7.000e+03 gram
Fuel left (U238)	: 9.930e+05 gram
Gamma energy released	: 2.117e-07 Mev
Beta Positive particles released	: 0.000e+00
Beta positive energy released	: 0.000e+00 Mev
Beta negative particles released	: 2.728e+19
Beta negative energy released	: 1.091e+19 Mev
Total energy released	: 1.091e+19 Mev

Summary report: 13

Fuel left (U235)	: 7.000e+03 gram

```
Fuel left (U238)                    : 9.930e+05 gram
Gamma energy released               : 6.541e-10 Mev
Beta Positive particles released    : 0.000e+00
Beta positive energy released       : 0.000e+00 Mev
Beta negative particles released    : 2.401e+19
Beta negative energy released       : 9.606e+18 Mev
Total energy released               : 9.606e+18 Mev
```

Summary report: 14

```
Fuel left (U235)                    : 7.000e+03 gram
Fuel left (U238)                    : 9.930e+05 gram
Gamma energy released               : 2.021e-12 Mev
Beta Positive particles released    : 0.000e+00
Beta positive energy released       : 0.000e+00 Mev
Beta negative particles released    : 2.126e+19
Beta negative energy released       : 8.505e+18 Mev
Total energy released               : 8.505e+18 Mev
```

Summary report: 15

```
Fuel left (U235)                    : 7.000e+03 gram
Fuel left (U238)                    : 9.930e+05 gram
Gamma energy released               : 6.243e-15 Mev
Beta Positive particles released    : 0.000e+00
Beta positive energy released       : 0.000e+00 Mev
Beta negative particles released    : 1.892e+19
Beta negative energy released       : 7.567e+18 Mev
Total energy released               : 7.567e+18 Mev
```

Summary report: 16

```
Fuel left (U235)                    : 7.000e+03 gram
Fuel left (U238)                    : 9.930e+05 gram
Gamma energy released               : 1.929e-17 Mev
Beta Positive particles released    : 0.000e+00
Beta positive energy released       : 0.000e+00 Mev
Beta negative particles released    : 1.691e+19
Beta negative energy released       : 6.762e+18 Mev
Total energy released               : 6.762e+18 Mev
```

Summary report: 17

```
Fuel left (U235)                    : 7.000e+03 gram
Fuel left (U238)                    : 9.930e+05 gram
Gamma energy released               : 5.959e-20 Mev
Beta Positive particles released    : 0.000e+00
Beta positive energy released       : 0.000e+00 Mev
Beta negative particles released    : 1.517e+19
Beta negative energy released       : 6.067e+18 Mev
```

```
Total energy released                    : 6.067e+18 Mev


Summary report: 18

Fuel left (U235)                         : 7.000e+03 gram
Fuel left (U238)                         : 9.930e+05 gram
Gamma energy released                    : 1.841e-22 Mev
Beta Positive particles released         : 0.000e+00
Beta positive energy released            : 0.000e+00 Mev
Beta negative particles released         : 1.366e+19
Beta negative energy released            : 5.464e+18 Mev
Total energy released                    : 5.464e+18 Mev


Summary report: 19

Fuel left (U235)                         : 7.000e+03 gram
Fuel left (U238)                         : 9.930e+05 gram
Gamma energy released                    : 5.689e-25 Mev
Beta Positive particles released         : 0.000e+00
Beta positive energy released            : 0.000e+00 Mev
Beta negative particles released         : 1.235e+19
Beta negative energy released            : 4.938e+18 Mev
Total energy released                    : 4.938e+18 Mev
\renewcommand{\baselinestretch}{0.65}
\begin{verbatim}


Summary report: 20

Fuel left (U235)                         : 7.000e+03 gram
Fuel left (U238)                         : 9.930e+05 gram
Gamma energy released                    : 1.758e-27 Mev
Beta Positive particles released         : 0.000e+00
Beta positive energy released            : 0.000e+00 Mev
Beta negative particles released         : 1.119e+19
Beta negative energy released            : 4.478e+18 Mev
Total energy released                    : 4.478e+18 Mev


Time for simulation: 4:4:1

Nuclide name        Amount(gram)


Cu72                 9.956e-24
Cu73                 9.614e-24
Cu74                 1.686e-08
Cu75                 8.657e-16
Cu76                 4.427e-11
Zn72                 8.457e-24
Zn73                 7.319e-24
Zn74                 5.859e-19
Zn75                 1.264e-16
Zn76                 5.209e-12
Zn77                 9.470e-20
```

Zn78	1.154e-15
Zn79	1.307e-11
Zn80	2.009e-08
Ga72	8.321e-24
Ga73	9.837e-24
Ga74	1.442e-19
Ga75	1.612e-15
Ga76	1.045e-12
Ga77	1.825e-20
Ga78	4.823e-16
Ga79	7.066e-12
Ga80	9.777e-09
Ga81	5.425e-13
Ga82	2.303e-08
Ga83	2.359e-06
Ge72	2.260e-10
Ge73	8.813e-10
Ge74	2.680e-09
Ge75	2.695e-14
Ge76	2.007e-07
Ge77	2.578e-20
Ge78	4.889e-14
Ge79	1.113e-12
Ge80	5.674e-10
Ge81	1.037e-13
Ge82	3.502e-09
Ge83	4.598e-07
Ge84	1.949e-11
As75	6.602e-08
As76	9.054e-24
As77	7.938e-21
As78	4.121e-13
As79	2.656e-12
As80	1.083e-09
As81	2.481e-14
As82	8.757e-10
As83	6.675e-08
As84	5.439e-12
As85	4.617e-17
As86	5.110e-10
As87	1.343e-08
Se76	1.906e-17
Se77	5.422e-07
Se78	1.442e-06
Se79	1.621e-05
Se80	4.846e-05
Se81	4.738e-14
Se82	1.223e-04
Se83	4.067e-08
Se84	1.425e-11
Se85	3.127e-18
Se86	3.262e-11
Se87	2.238e-09
Se88	2.214e-13
Se89	5.663e-06
Se90	1.975e-03
Se91	4.899e-05
Br81	1.355e-05
Br82	2.035e-18
Br83	4.340e-07
Br84	1.536e-12
Br85	1.191e-16
Br86	8.961e-12
Br87	2.274e-10

Br88	2.229e-14
Br89	5.863e-07
Br90	1.342e-15
Br91	4.805e-05
Br92	1.948e-05
Br93	2.241e-03
Br94	4.768e-04
Kr82	1.462e-09
Kr83	2.037e-04
Kr84	3.917e-04
Kr85	3.930e-17
Kr86	1.484e-04
Kr87	2.701e-08
Kr88	2.728e-13
Kr89	9.351e-07
Kr90	8.390e-17
Kr91	3.113e-06
Kr92	4.416e-06
Kr93	7.862e-12
Kr94	1.287e-04
Kr95	4.932e-08
Kr96	2.288e-03
Kr97	1.145e-04
Rb85	9.854e-05
Rb86	9.326e-24
Rb87	1.034e-03
Rb88	4.782e-14
Rb89	1.965e-07
Rb90	3.871e-15
Rb91	4.637e-07
Rb92	1.963e-06
Rb93	2.224e-12
Rb94	1.081e-05
Rb95	2.674e-03
Rb96	2.977e-05
Rb97	3.166e-04
Rb98	4.924e-04
Rb99	1.016e-03
Rb100	2.366e-04
Rb101	4.083e-04
Rb102	1.548e-05
Sr86	1.065e-14
Sr88	2.765e-04
Sr89	6.039e-08
Sr90	3.700e-16
Sr91	2.839e-07
Sr92	3.710e-06
Sr93	2.130e-12
Sr94	2.818e-05
Sr95	4.017e-07
Sr96	6.880e-06
Sr97	1.862e-04
Sr98	9.704e-05
Sr99	2.808e-04
Sr100	2.234e-04
Sr101	4.074e-04
Sr102	1.565e-04
Y89	1.968e-03
Y90	1.398e-14
Y91	4.631e-07
Y92	3.142e-06
Y93	1.769e-12
Y94	1.905e-06
Y95	9.793e-07

Y96	1.215e-06
Y97	2.156e-05
Y98	1.305e-04
Y99	5.339e-05
Y100	7.784e-05
Y101	1.479e-04
Y102	4.638e-05
Zr90	4.595e-04
Zr91	2.377e-03
Zr92	2.554e-03
Zr93	5.215e-04
Zr94	2.680e-03
Zr95	1.576e-07
Zr96	4.945e-04
Zr97	4.864e-06
Zr98	2.516e-06
Zr99	3.658e-05
Zr100	8.180e-06
Zr101	3.200e-05
Zr102	6.042e-06
Zr103	3.602e-12
Zr104	9.109e-12
Nb95	2.885e-07
Nb96	1.089e-28
Nb97	7.441e-05
Nb98	2.765e-05
Nb99	5.385e-06
Nb100	4.325e-05
Nb101	9.617e-06
Nb102	1.529e-05
Nb103	2.202e-11
Nb104	3.036e-12
Nb105	4.443e-23
Nb106	3.282e-11
Mo95	2.035e-04
Mo96	1.804e-12
Mo97	2.439e-03
Mo98	1.891e-03
Mo99	2.998e-05
Mo100	2.829e-03
Mo101	4.713e-06
Mo102	1.783e-06
Mo103	1.019e-09
Mo104	1.075e-09
Mo105	7.636e-24
Mo106	4.435e-12
Mo107	9.798e-24
Mo108	4.430e-06
Tc99	1.394e-03
Tc100	9.054e-24
Tc101	4.885e-06
Tc102	3.921e-06
Tc103	2.177e-11
Tc104	6.247e-11
Tc105	3.205e-23
Tc106	1.064e-12
Tc107	9.219e-24
Tc108	9.817e-24
Tc109	1.466e-15
Tc110	2.204e-11
Tc111	2.582e-08
Ru100	5.724e-14
Ru101	1.833e-03
Ru102	2.190e-03

Ru103	3.079e-11
Ru104	1.721e-04
Ru105	1.647e-22
Ru106	4.651e-10
Ru107	9.725e-24
Ru108	9.978e-24
Ru109	6.200e-17
Ru110	1.291e-12
Ru111	6.454e-09
Ru112	9.689e-24
Ru113	5.340e-23
Rh103	2.747e-04
Rh104	9.079e-24
Rh105	2.262e-23
Rh106	1.638e-11
Rh107	7.871e-24
Rh108	8.144e-24
Rh109	4.681e-14
Rh110	8.530e-12
Rh111	9.048e-10
Rh112	9.489e-24
Rh113	8.489e-12
Rh114	2.363e-18
Rh115	9.580e-13
Rh116	7.332e-11
Pd104	8.647e-13
Pd105	8.873e-05
Pd106	3.826e-05
Pd107	1.366e-05
Pd108	7.041e-07
Pd109	5.139e-15
Pd110	2.421e-06
Pd111	4.309e-10
Pd112	7.747e-24
Pd113	1.032e-11
Pd114	6.254e-18
Pd115	2.061e-14
Pd116	4.277e-12
Pd117	9.796e-24
Pd118	3.402e-21
Ag109	4.079e-07
Ag110	7.326e-24
Ag111	1.406e-09
Ag112	9.620e-24
Ag113	3.848e-12
Ag114	5.539e-18
Ag115	5.096e-14
Ag116	1.990e-12
Ag117	6.690e-14
Ag118	5.104e-21
Ag119	3.614e-20
Ag120	4.783e-14
Ag121	3.105e-11
Ag122	3.263e-09
Ag123	1.764e-07
Ag124	7.057e-07
Cd110	2.867e-15
Cd111	1.821e-06
Cd112	1.282e-06
Cd113	1.492e-06
Cd114	1.789e-07

Cd115	8.233e-13
Cd116	1.327e-06
Cd117	6.427e-14
Cd118	4.262e-22
Cd119	1.286e-19
Cd120	1.187e-15
Cd121	1.904e-12
Cd122	3.855e-10
Cd123	3.072e-08
Cd124	1.522e-07
Cd125	2.356e-09
Cd126	4.643e-08
Cd127	1.407e-07
Cd128	1.591e-06
Cd129	4.440e-06
Cd130	2.926e-06
In115	9.718e-07
In116	9.928e-24
In117	3.741e-15
In118	8.245e-21
In119	1.576e-18
In120	3.224e-14
In121	1.157e-12
In122	2.173e-09
In123	1.128e-08
In124	6.376e-08
In125	9.536e-10
In126	2.175e-08
In127	7.608e-08
In128	8.616e-07
In129	3.456e-06
In130	1.004e-05
In131	9.355e-06
In132	4.404e-05
In133	1.073e-05
Sn116	3.833e-14
Sn117	1.133e-06
Sn118	1.350e-06
Sn119	1.868e-07
Sn120	1.373e-06
Sn121	7.586e-11
Sn122	9.040e-06
Sn123	5.253e-10
Sn124	1.700e-05
Sn125	2.515e-10
Sn126	7.388e-05
Sn127	5.044e-08
Sn128	1.172e-08
Sn129	1.030e-06
Sn130	8.399e-07
Sn131	6.765e-08
Sn132	2.213e-07
Sn133	1.534e-06
Sn134	3.897e-10
Sb121	1.704e-06
Sb122	9.942e-24
Sb123	1.527e-05
Sb124	8.330e-24
Sb125	1.165e-09
Sb126	9.905e-24
Sb127	3.109e-08
Sb128	7.856e-08
Sb129	6.009e-07
Sb130	8.142e-08

Sb131	1.161e-07
Sb132	2.213e-06
Sb133	9.519e-07
Sb134	9.876e-08
Sb135	3.254e-14
Sb136	7.963e-09
Te122	1.081e-10
Te124	2.142e-10
Te125	3.439e-05
Te126	8.402e-08
Te127	1.326e-08
Te128	5.200e-04
Te129	3.067e-06
Te130	2.075e-03
Te131	1.080e-07
Te132	3.038e-06
Te133	1.947e-07
Te134	1.917e-09
Te135	3.218e-15
Te136	3.915e-10
Te137	2.464e-18
Te138	2.590e-12
I126	1.142e-33
I127	2.416e-04
I128	7.891e-24
I129	1.258e-03
I130	9.220e-24
I131	3.481e-07
I132	4.761e-06
I133	1.171e-07
I134	1.555e-09
I135	1.258e-14
I136	1.203e-08
I137	2.799e-19
I138	7.110e-13
I139	3.924e-17
I140	5.001e-09
I141	1.473e-06
I142	4.671e-05
Xe128	5.348e-09
Xe129	1.489e-14
Xe130	1.988e-06
Xe131	3.063e-03
Xe132	4.753e-03
Xe133	4.811e-07
Xe134	4.770e-03
Xe135	1.119e-14
Xe136	4.593e-03
Xe137	5.196e-18
Xe138	3.639e-13
Xe139	2.413e-18
Xe140	3.376e-10
Xe141	5.221e-07
Xe142	9.159e-06
Xe143	9.891e-06
Xe144	2.709e-11
Xe145	2.116e-10
Cs131	4.063e-32
Cs132	9.694e-24
Cs133	5.811e-03
Cs134	1.624e-20
Cs135	4.629e-03
Cs136	8.867e-24
Cs137	7.173e-19

Cs138	1.666e-13
Cs139	1.368e-17
Cs140	2.344e-08
Cs141	3.673e-08
Cs142	6.984e-06
Cs143	2.005e-06
Cs144	4.093e-09
Cs145	1.442e-07
Cs146	3.050e-06
Cs147	8.442e-06
Cs148	2.477e-06
Ba132	2.789e-14
Ba134	4.890e-08
Ba135	4.753e-08
Ba136	4.455e-06
Ba137	4.287e-03
Ba138	4.808e-03
Ba139	4.392e-11
Ba140	2.091e-09
Ba141	5.123e-08
Ba142	1.197e-06
Ba143	2.549e-07
Ba144	3.976e-10
Ba145	2.494e-08
Ba146	5.230e-07
Ba147	2.886e-06
Ba148	7.758e-07
Ba149	5.349e-07
La136	6.057e-32
La137	5.806e-14
La138	4.920e-08
La139	4.025e-03
La140	3.281e-08
La141	2.717e-07
La142	9.561e-06
La143	2.642e-07
La144	1.142e-10
La145	4.262e-09
La146	1.925e-07
La147	6.788e-07
La148	5.353e-07
La149	2.603e-07
Ce139	1.554e-28
Ce140	5.014e-03
Ce141	3.307e-08
Ce142	5.024e-03
Ce143	3.449e-06
Ce144	3.525e-11
Ce145	4.241e-08
Ce146	9.202e-08
Ce147	4.893e-08
Ce148	1.055e-08
Ce149	5.916e-08
Ce150	9.960e-24
Ce151	1.522e-10
Ce152	1.266e-23
Pr141	5.051e-03
Pr142	8.540e-24
Pr143	3.587e-07
Pr144	2.324e-10
Pr145	2.360e-08
Pr146	5.202e-08
Pr147	2.080e-07
Pr148	2.894e-07

Pr149	1.586e-07
Pr150	9.913e-24
Pr151	7.111e-12
Pr152	2.990e-22
Pr153	9.993e-24
Pr154	2.240e-19
Nd142	1.773e-11
Nd143	4.265e-03
Nd144	3.081e-03
Nd145	2.421e-03
Nd146	1.784e-03
Nd147	2.592e-07
Nd148	8.855e-04
Nd149	2.684e-07
Nd150	2.913e-04
Nd151	1.397e-11
Nd152	1.148e-22
Nd153	6.288e-24
Nd154	2.183e-20
Nd155	9.953e-24
Nd156	9.308e-24
Pm146	2.364e-32
Pm147	1.189e-06
Pm148	9.374e-24
Pm149	2.495e-07
Pm150	1.805e-23
Pm151	8.741e-10
Pm152	1.263e-21
Pm153	9.840e-24
Pm154	4.558e-15
Pm155	5.009e-24
Pm156	7.468e-24
Pm157	8.880e-24
Pm158	9.902e-24
Sm147	1.323e-03
Sm148	8.952e-09
Sm149	5.281e-04
Sm150	3.312e-07
Sm151	1.162e-11
Sm152	1.120e-04
Sm153	3.285e-16
Sm154	2.298e-05
Sm155	6.987e-24
Sm156	9.484e-24
Sm157	9.003e-24
Sm158	9.743e-24
Sm159	9.686e-24
Eu151	1.887e-04
Eu152	8.859e-24
Eu153	5.280e-05
Eu154	8.914e-24
Eu155	9.439e-24
Eu156	8.339e-24
Eu157	9.961e-24
Eu158	4.490e-24
Eu159	8.651e-24
Gd152	2.129e-10
Gd153	1.278e-16
Gd154	2.442e-21
Gd155	1.077e-05
Gd156	4.510e-06
Gd157	4.575e-07
Gd158	7.368e-07
Gd159	8.494e-24

Tb157	3.272e-17
Tb159	4.199e-08
Tb160	9.748e-24
Dy158	8.802e-12
Dy159	3.457e-18
Dy160	2.791e-15
Dy161	3.098e-22
Dy162	2.948e-28
U235	7.000e+03
U236	2.216e-02
U237	9.522e-24
U238	9.930e+05
U239	8.521e-24
Np237	1.709e-09
Np238	9.900e-24
Np239	5.286e-16
Pu238	1.197e-17
Pu239	8.617e-02
Pu240	5.454e-08
Pu241	9.884e-24
Pu242	3.012e-20
Pu243	2.560e-27
Am241	6.188e-14
Am242	8.595e-24
Cm242	8.133e-20
Cm243	2.207e-27

VITA

Candidate's full name	: Sri Hartati
Place and date of birth	: Surakarta, Indonesia
	September 21, 1961
Permanent address	: Sidoluhur 57, Laweyan Surakarta, 57148
	Indonesia
	Phone: (271) 712 248
Schools attended	: Surakarta Junior High School I
	Surakarta, Indonesia, 1974 - 1976
	Surakarta Senior High School I
	Surakarta, Indonesia, 1976 - 1980
Universities attended	: Gadjah Mada University
	Yogyakarta, Indonesia
	Dra. (Electronics) 1980 - 1986
	University of New Brunswick
	Fredericton, N.B., Canada
	M.Sc. (CS) 1988 - 1990
	University of New Brunswick
	Fredericton, N.B., Canada
	PhD. (CS) 1992 - now

Publications:

1. Hartati, S., Nickerson, B. G., and DeMille, R. G. "Reasoning About Nuclear Physics Processes", *Proceedings of the 7th IEEE International Conference on*

Tools with Artificial Intelligence, Washington, D.C, November 5-8, 1995, pp.228-235.

2. Hartati, S., Nickerson, B.G., and DeMille, G.R. "A model for Simulation of Nuclear Physics Processes", *International Journal of Modelling and Simulation*, accepted September, 1995, 24 pages (in press).

3. Hartati, S. "A model for Computer Reasoning about Nuclear Physics Processes", *Bianglala Journal, Indonesian Scientific Journal*, vol.1, no.3, pp.9-16.

4. Hartati, S. "Bahasa Pemrograman Komputer Untuk Fisika Nuklir dan Terapannya di Indonesia" ("A Programming Language for Nuclear Physics Processes and Its Applications in Indonesia"), *Proceedings of the 4th Seminar, Communication Forum of the Indonesian Students in Canada*, Fredericton, N.B., August 24-25, 1995, pp. 127-142.

5. Harjoko, A. and Hartati, S. "Pengembangan Basis Data Gambar Digital untuk Pertahanan dan Keamanan di Indonesia" ("Design of Digital Picture Database for National Defense and Security of Indonesia"), *Proceedings of the 4th Seminar, Communication Forum of the Indonesian Students in Canada*, Fredericton, N.B., August 24-25, 1995, pp. 115-126.

6. Hartati, S and Nickerson, B.G. "An Efficient Computer Representation of Nuclide Data", *Computational Materials Science*, submitted, June, 1995, 18 pages.

7. Hartati, S. "Nuclear Process Theory: A Symbolic Model for Representing Nuclear Physics Processes", Technical Report TR94-087, Revision 2, Faculty of Computer Science, University of New Brunswick, May 4, 1995.

8. Hartati, S. "NPT : a model for computer reasoning about nuclear physics processes", *Third Annual Graduate Student Association Conference on Student Research*, Fredericton, NB, April 19, 1995, p.9.

9. Hartati, S and Nickerson, B. G. "Nuclear Process Theory", *Proceedings of the 11th IEEE Conference on AI Applications*, Los Angeles, CA, February 19 - 22, 1995, pp.340-346.

10. Hartati, S and Nickerson, B. G. "A Symbolic Model for Representing Nuclear Physics Processes", *Proceedings of the APICS Annual Computer Science Conference*, Wolfville, Nova Scotia October 29, 1994, pp.99-108.

11. Hartati, S. "Reasoning about Physical Systems in Artificial Intelligence", *Technical Report TR93-080*, Faculty of Computer Science, University of New Brunswick, 1993.

12. Hartati, S. "Representing two dimensional objects using orientation adaptive quadtrees (OAQs)", *Jurnal Jurusan Fisika, Universitas Gajah Mada*, vol.2, no.5, pp.39-59, 1991.

13. Hartati, S. *Orientation Adaptive Quadtrees*, Master Thesis, Faculty of Computer Science, University of New Brunswick, 1990.

14. Nickerson, G and Hartati, S. "Constructing Orientation Adaptive Quadtrees", *Proceedings Graphics Interface 90*, Halifax, May 1990, pp.190-195.

15. Hartati, S. *Simulasi elektronik mesin penjual gula-gula* (*Electronic simulation of candies vending machine*), Bachelors Thesis, Department of Electronics, Faculty of Mathematics and Natural Sciences, Gajah Mada University, Yogyakarta, 1986.