ABSTRACT

OLIVEIRA DOS SANTOS, BARBARA. Optimization and Acceleration of the Nodal Expansion Method (NEM) and its Multi-Physics Coupling with the Sub-channel Thermal-Hydraulic/Fuel Modeling Code CTF/CTFFuel. (Under the direction of Dr. Maria N. Avramova and Dr. Agustin Abarca).

Given the consequences of economic and environmental difficulties, it is projected that current commercial nuclear plants will operate much beyond their initial license period. However, this is achievable only if these facilities continue to run safely and economically. The ability to conduct a complete safety analysis of design or beyond design operational events and support plant enhancements, such as power uprates, is critical for achieving this goal.

Because of the rising demand for energy, it is preferable to operate a nuclear reactor with longer cycles and higher power generation capacity factor. This condition necessitates the development of improved fuel elements with greater enrichment and accident tolerance features, for instance. These new initiatives may alter some of the authorized safety parameters for the operation of a nuclear reactor. Consequently, safety mechanisms and specific operation characteristics must be evaluated in depth to assure that the reactor will stay safe even under the most adverse conditions.

As the demand for ever-increasing levels of computational capacity grew, the designers of computing systems began to consider ways in which multiple of the computing machines they already possessed could be combined into a single system. Parallel computers are extremely widespread and are put to great use for the processing of complicated computations. The development of algorithms and computer codes that can use the capabilities of the existing hardware in order to tackle larger problems in a shorter amount of time is one of the challenges presented by parallel computers. Open Multi-Processing (OpenMP) is an Application

Programming Interface (API) that supports multi-platform shared-memory multiprocessing programming in C++, and FORTRAN.

The past decades have been defined by astounding technological and computational advancements. The most significant advancement is to the processing speed of computer programs and memory capacity. This fact had a significant impact on nuclear simulations, as they employ many computer codes, allowing the coupling of codes for more precise multi-physics design, operation and safety analysis calculations with real feedback effects considered.

To ensure that safety limitations are not exceeded, nuclear reactors must be analyzed beforehand using computer programs to study its thermo-mechanical (fuel performance), thermal-hydraulics, neutronics, and coupled multi-physics behavior.

In this context, this PhD dissertation presents an efficient multi-physics coupling between the sub-channel code CTF, its nuclear fuel rod solver CTFFuel, and the nodal neutron diffusion code NEM. CTF/CTFFuel, co-developed by North Carolina State University (NCSU) and Oak Ridge National Laboratory, and NEM (Nodal Expansion Method) are being maintained by the Reactor Dynamics and Fuel Modeling Group (RDFMG) at NCSU. The coupling is performed using the well know Picard Iteration (PI) method. However, PI is susceptible to over-solving.

Because there are several single-physics solvers interacting with one other and modifying the solutions within a multi-physics solution process, the management of the over-solving problem in multi-physics coupling is complex. The performed literature review demonstrates that exact answers can be reached in approximate approaches by relaxing tolerances in the single-physics solvers. The inexact method known as Residual Balance Method (RBM) has been explored in this work, resulting in better acceleration of the convergence rate of the coupled codes multi-physics system.

Several important contributions of this study can be outlined. *The first* enhancement is the improved acceleration of NEM by implementing the FORTRAN ‘Do Concurrent (DC)’ using OpenMP. The directives on a DC structure within OpenMP are investigated. *The second* contribution is the improvement of NEM standalone code using the Coarse Mesh Rebalancing (CMR) approach, which resulted in acceleration of the neutronics code. *The third* enhancement involves modifying the PI method to increase its efficiency by employing RBM, resulting in faster convergence of the coupled codes system. Finally, the developed multi-physics codes system CTF/CTFFuel/NEM is used to accurately simulate nuclear reactor cores, including pin-wise and assembly-wise models i.e., at two different levels (scales) of spatial resolution. The Pressurized Water Reactor (PWR) Main Steam Line Break (MSLB) Benchmark and the coupled PARCS/CTF code system are used to verify CTF/CTFFuel/NEM.

Optimization and Acceleration of the Nodal Expansion Method (NEM) and its Multi-Physics Coupling with the Sub-channel Thermal-Hydraulic/Fuel Modeling Code CTF/CTFFuel

by
Barbara Oliveira dos Santos

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Nuclear Engineering

Raleigh, North Carolina
2022

APPROVED BY:

_____	_____
Dr. Maria N. Avramova Co-Chair of Advisory Committee	Dr. Agustin Abarca Co-Chair of Advisory Committee
_____	_____
Dr. Kostadin Ivanov	Dr. Christopher Healey

DEDICATION

To my parents Ivan (in memoriam) and Mariluci who gave up many of their dreams so that I could fulfill mine (*Aos meus pais Ivan (in memoriam) e Mariluci que abriram mão de muitos de seus sonhos para que eu pudesse realizar os meus*).

To my daughter, Sarah, who taught me about tolerance and respect.

To Ken Schultz, my husband, friend, and partner who has always supported me.

BIOGRAPHY

The author was born in Brazil. She obtained her bachelor's degree in physics in 1984 and her master's degree in Nuclear Engineering in 1990 from the Federal University of Rio de Janeiro. In 1989, he started working on PWR nuclear plants in Brazil, where she remained until 2018. In 2018 she moved to the United States, and in 2019 she started pursuing her PhD in multi-physics simulation for the design and safety analysis of nuclear reactors at NCSU, North Carolina, under the supervision of Dr. Maria Avramova, Dr. Kostadin Ivanov, Dr. Agustin Abarca and Dr. Muhammad Ramzy Altahhan.

In the area of the Safety Analysis, she developed studies and technical reports that served as support for design changes, such as “Requirements for Safe Shutdown after Fire” for the Angra 1 NPP, development, implementation and commissioning of the Environmental Control System for the Angra 2 NPP (evaluation of radiation releases in case of accidents), and the study of core damage evaluation after severe accident – taking into account that this last study served as the foundation for the evaluation of decommissioning of the Angra 1 Post Accident Sampling System. In 2005, she left the design area, and she joined the operating area at Angra NPP site, becoming part of the reactor physics group until 2018.

ACKNOWLEDGMENTS

First of all, to God for giving me strength and enthusiasm to continue in this walking.

To my parents Ivan (in memoriam) and Mariluci, inexhaustible sources of love, affection, and unconditional dedication to their two daughters.

To my husband Ken Schultz for his patience and understanding.

To my sister, who encourage me unmeasured for my success and see in me qualities that, most of the time, I don't have.

To CNPq for, in 2016, sponsor me to come to NCSU and meet this fantastic group of researchers.

Dr. Maria was a model of caring and guidance. I have been privileged to be able to work on this opportunity of a PhD project under her supervision and tutelage.

Dr. Kostadin Ivanov for his support in carrying out this work. I am also grateful and thankful to my PhD committee: Dr. Kostadin Ivanov, Dr. Maria Avramova, Dr. Agustin Abarca and Dr. Christopher Healey.

To my advisor Dr. Agustin Abarca and my mentor Dr. Muhammad Ramzy, I am deeply grateful for your receptivity, friendship, and constant help, which made the development of this thesis an extremely pleasant, and relaxed task.

In different parts of the PhD journey, I met many people that helped advance my career and to whom I confer my thanks. Of those individuals, I especially mention Dr. Mohamed Bourham.

To all the friends who accompanied me during this journey, encouraging and helping me in some way. I sincerely thank my RDFMG colleagues. I would like to thank my friends Khaldoon, Ghada and Rofhiwa for their encouragement and for great times we had together.

I would like to thank the Nuclear Engineering department's staff for their help with many bureaucratic works facilitating the PhD process many times, especially Mr. Robert Green and Mr. Mario Milev.

TABLE OF CONTENTS

LIST OF TABLES viii
LIST OF FIGURES ix

CHAPTER 1 INTRODUCTION AND THESIS STRUCTURE 1
1.1 Thesis's Foundation 1
1.2 Thesis Outline and Layout 5

CHAPTER 2 LITERATURE REVIEW 7
2.1 Multi-physics Codes System 7
2.2 Multi-physics Coupling Methods 11
2.2.1 External and Internal Coupling 12
2.2.2 Jacobian-Free Newton-Krylov Methods 13
2.2.3 Approximate Block Newton Methods 14
2.2.4 Picard Iteration Method 14
2.3 Acceleration Methods for Coupled Codes 15
2.3.1 Solution Interruption 16
2.3.2 Alternating Nonlinear Method 18
2.3.3 Relaxed Relative Tolerance 18
2.3.4 Residual Balance 20
2.4 Standalone NEM Acceleration Methods 21
2.4.1 Coarse Mesh Finite Difference (CMFD) Method 21
2.4.2 CMR with Wielandt-shift and Asymptotic Extrapolation 22
2.4.3 Multilevel Coarse Mesh Rebalance 24
2.4.4 Parallelization and System-based Optimizations 24
2.5 Conclusions and Summary 29

CHAPTER 3 THEORY DEVELOPMENT 31
3.1 Introduction 31
3.2 Acceleration/Optimization Methods in NEM 31
3.2.1 CMR Method with Wielandt-shift and Asymptotic Extrapolation 33
3.2.2 NEM Acceleration Methods Stabilization and Optimization 37

3.2.3 Techniques of Parallelization 44
3.3 Residual Balance Method Improvements 54
3.3.1 Residual Balance Method 54
3.3.2 Adaptive Residual Balance Method 56
3.3.3 Coupled Convergence Methods 57

CHAPTER 4 CTF/NEM COUPLED CODE 59
4.1 Message Passing Information – Communication Interface 59
4.2 Temporal Coupling 60
4.3 Spatial Coupling 62

CHAPTER 5 RESULTS AND DISCUSSIONS 66
5.1 Code Verification and Validation 66
5.2 TMI Benchmark Assembly-wise Analysis 67
5.2.1 Introduction 67
5.2.2 Model Description 68
5.2.3 NEM Standalone Analysis 72
5.2.4 Coupled Analysis 76
5.3 C5G7-TD Analysis 90
5.3.1 Model Description 90
5.3.2 Cross-Sections 94
5.3.3 Standalone Analysis 98
5.3.4 Coupled Analysis 104

CHAPTER 6 CONCLUSIONS AND FUTURE WORKS 108
6.1 Summary and Conclusions 108
6.2 Recommendations for Future Work 110

REFERENCES 114
APPENDIX 121

LIST OF TABLES

Table 1: Termination Criteria for Different Coupled Convergence Methods 58

Table 2: Steps to be Followed in the Process of Verification of the Codes as well as the Coupling System 67

Table 3: TMI Model - Fuel Assembly Geometry Data 69

Table 4: TMI Model – Assembly Types 70

Table 5: Initial Conditions for TMI Model 71

Table 6: Total Coupled-Simulation Time for NEM and CTF with OpenMP and CMR. 77

Table 7: Total and Solver Simulation Times in Seconds for NEM with Different Acceleration Measures. 78

Table 8: Number Coupled Global Iterations and Run Time for the Coupled Convergence. 87

Table 9: Fuel Assembly/Core Data. 93

Table 10: Spacer Grid Positions, Loss Coefficients and Height of the Grids. 94

Table 11: The Reactor States Involved in Generating the Cross-Section Library 96

Table 12: The Energy Group Structure used for Collapsing the Cross-Sections. 96

Table 13: The Cross-Sections Sets found inside the nemtab File 97

LIST OF FIGURES

Figure 2-1: General Operator Splitting Scheme 12

Figure 2-2: Description of the PI Using Two Solvers 15

Figure 2-3: Solution Interruption Method 17

Figure 2-4: Alternating Nonlinear Method 19

Figure 3-1: Help Document for Installing the NEM Code Invoked by "./install_NEM.sh -h" in the NEM Root Directory 33

Figure 3-2: Example of gprof's Flat Profile Result Output File Generated by Running the NEM Code on the TMI Benchmark Problem 40

Figure 3-3: Loop-Based Profile of the NEM Program 42

Figure 3-4: Loop-611 Time Coverage of the Total Wall Time and its contents 43

Figure 3-5: Expert Summary of the Loops that Require to be Revised and Considered for Optimization 43

Figure 3-6: Shared Memory and Distributed Memory 45

Figure 3-7: Serial Computing Example 46

Figure 3-8: Parallel Computing Example 47

Figure 3-9: Model of Forks and Joins used in OpenMP 52

Figure 3-10: Example of Inner Loop Iteration that Solves the Outer Current Equation in NEM Code 53

Figure 4-1: General Scheme of the Server/Client Based MPI Coupling Interface 60

Figure 4-2: Synchronized Communication Representation of an MPI_SEND - MPI_RECV Pair Block 62

Figure 4-3: Spatial Coupling Mapping “Table1.map” and “Table2.map” file Contents Visualized....64
Figure 5-1: Radial (a) and Axial (b) Nodalization of the TMI CTF/NEM Model 71
Figure 5-2: Coarse Mesh used for the TMI Benchmark Problem 73
Figure 5-3: Time Comparison with Different Accelerations Measures for NEM Standalone Code Using the TMI Benchmark Problem....73
Figure 5-4: Time Comparison Between Different Values for OpenMP Threads Using the TMI Benchmark Problem 74
Figure 5-5: Time Comparison with Enhanced CMR Solver for NEM Standalone Code Using the TMI Benchmark Problem 75
Figure 5-6: Applying NEM Acceleration Measures to the Simulation of the TMI-Benchmark Coupled Case.....76
Figure 5-7: Performance of the Different Acceleration Measures for the Coupled-Simulation of the TMI-Benchmark Problem....79
Figure 5-8: 2D Radial Power Profile Computed by CTF/NEM. 80
Figure 5-9: 2D Radial Power Profile Computed by CTF/PARCS. 80
Figure 5-10: Relative Difference for the 2D Radial Power Profile Between both Codes. 81
Figure 5-11: 1D axial Power Profile Comparison Between both Codes....81
Figure 5-12: 2D Radial Fuel Effective Temperature Computed by CTF/NEM. 82
Figure 5-13: 2D Radial Fuel Effective Temperature Computed by CTF/PARCS 83
Figure 5-14: Absolute Relative Difference for the Radial Fuel Effective Temperature Between Both Codes 83
Figure 5-15: Axial Fuel Effective Temperature Computed by CTF/NEM and CTF/PARCS. 84

Figure 5-16: 2D Radial Coolant Density (g/cm3) Computed by CTF/NEM. 84
Figure 5-17: 2D Radial Coolant Density (g/cm3) Computed by CTF/PARCS. 85
Figure 5-18: Relative Difference for the Radial Coolant Density Between Both Codes.....85
Figure 5-19: Axial Coolant Density (g/cm3) Computed by CTF/NEM and CTF/PARCS.....86
Figure 5-20: Evolution of the Residual During the Convergence with the Solution Interruption Method in S2. 88
Figure 5-21: Evolution of the Residual during the Convergence with the Solution Interruption of S1 and S2 Method.....88
Figure 5-22: Evolution of the Residual during the Convergence with the Residual Balance Method. 89
Figure 5-23: Radial and Axial Nodalization of the C5G7 CTF/NEM Model.....91
Figure 5-24: Spacer Grids Axial Positions and Axial Nodes Equivalence to Elevation. 92
Figure 5-25: 2D Configuration for C5G7-TD Problem. 93
Figure 5-26: C5G7 Pin-Wise Geometry.....100
Figure 5-27: The Radially Collapsed 1D Axial Power Profile for Each Z-Axis Level in the C5G7-TD Benchmark. 101
Figure 5-28: The Axially Collapsed 2D Radial Power in Each Pin in the C5G7-TD Benchmark. 101
Figure 5-29: Acceleration by Using a Modern CPU Architecture on the New RDFMG Cluster Nodes. 103
Figure 5-30: Acceleration of the C5G7 Benchmark Simulation by Using Both Several OpenMP Threads and CMR.....104

Figure 5-31: The Absolute Difference in 2D Radial Power in Each Pin in the C5G7-TD Benchmark for a Simulation with CMR and with no CMR. 105
Figure 5-32: CTF/NEM Radial Power Distribution. 106
Figure 5-33: CTF/NEM Axial Power Distribution. 106
Figure 5-34: Radial Coolant Outlet Temperature Determined by the CTF/NEM Code. 107
Figure 5-35: CTF/NEM Axial Coolant Outlet Temperature. 107
Figure 5-36: Radial Core Outlet Coolant Density Determined by the CTF/NEM Code. 108
Figure 5-37: Effective Fuel Temperature Determined by the CTF/NEM Code 108

NOMENCLATURE

ABN	Approximation Block Newton
API	Application Interface
ARB	Architecture Review Board
BWR	Boiled Water Reactor
CMFD	Coarse-Mesh Finite Difference
CMR	Coarse Mesh Rebalance
COBRA-TF	Coolant-Boiling in Rod Arrays – Two Fluids
DC	Do Concurrent
HBHE	High Burnup High Energy
HPC	High-Performance Computing
JFNK	Jacobian-Free Newton-Krylov
MG	Multigrid Method
ML-CMR	Multi-Level Coarse Mesh Rebalance
MPI	Message Passing Information
NCSU	North Carolina State University
NEA	Nuclear Energy Agency
NEM	Nodal Expansion Method
NPPs	Nuclear Power Plants
NSC	Nuclear Science Committee
OEDC	Organization for Economic Co-operation and Development
OpenACC	Open Accelerators
OpenMP	Open Multi-Processing
OS	Operator Splitting
PARCS	Purdue Advanced Reactor Core Simulator
PDE	Partial Differential Equation
PI	Picard Iteration
PWR	Pressurized Water Reactor
RBM	Residual Balance Method

RDFMG	Reactor Dynamic and Fuel Modeling Group
SMP	Symmetric Multi-Processing

CHAPTER 1: INTRODUCTION AND THESIS STRUCTURE

1.1 Thesis's Foundation

Users of computer models in decision making have increasingly high expectations of those models as a direct result of the continued improvement in both the speed and capacity of computer systems. Some people in today's world look to computer-based predictions to obtain essential quantitative information on events that influence the safety, health, and well-being of a large portion of the human population and many nations, as well as events that influence the success of major businesses. In other words, some people look to computer-based predictions to find out how to protect a large portion of the human population (Babuska & Oden, 2004).

In the view of current economic and environmental challenges, it is projected that the current commercial Nuclear Power Plants (NPPs) will operate much beyond their initial license period (using lifetime extensions) at higher power and with a better fuel utilization. These objectives are achievable only if the NPPs continue to operate safely and economically within an envelope of reduced margins. The ability to conduct a complete safety analysis of design or beyond design operational events and to support plant enhancements, such as life extension, power uprates, and utilization of high burnup high enrichment (HBHE) fuel, is critical for achieving this goal.

During the last few decades, we have seen significant developments in computing hardware and software and associated technology. The primary advancements are related to the processing speed of computer programs and the amount of memory available for storing data, i.e., computation efficiency. This fact had a significant influence on the NPPs simulations, as they employ extensive numerical simulations, allowing for the multi-physics coupling of codes for more accurate safety analysis calculations.

To ensure that safety limits will not be exceeded in nuclear reactors, they must be analyzed and evaluated using computer programs to model and simulate their fuel, thermal-hydraulics, neutronics, and multi-physics behavior.

Simulations of complex scenarios involving instabilities in power reactors were improved using coupled thermal-hydraulic and neutron kinetics systems codes. As a result, there is an obvious need for multi-physics coupled codes that incorporate modeling of as many physical processes occurring in nuclear reactors as possible. Another desired capability is to have more efficient coupling algorithms, particularly important when the number of coupled physics phenomena modeled increases. A multi-physics analysis that integrates efficient input from each physics process produces the most accurate predictions compared with the real system behavior.

The use of computational methods is of great importance to meet the requirements to provide an accurate solution with a reasonable CPU time.

Neutron physics calculations performed on nuclear reactors continue to make use of the so-called nodal methods (type of finite volume techniques) including Nodal Expansion Methods even in modern times (e.g., eigenvalue calculation, search mechanisms, loading pattern optimization, reactor operation, safety analysis, etc.). This is because these computations are performed quickly, and the results that are obtained are accurate, as compared to measede data, which are subject to uncertainties (depending on the approximation used). The first improvement and contribution in this thesis comes from acceleration of the NEM code by using the enhanced and stabilized CMR (Coarse Mesh Rebalancing) method, resulting in global acceleration of the overall coupled-code multi-physics system.

Recent advancements in computer architecture have made it possible for a single processor to interleave the execution of numerous instruction streams (Chapman Barbara et al., n.d.). The

computer might, for example, add two values from one set of instructions while also retrieving a value from memory that is required to carry out an action in another set of instructions. The method in question is referred to as multicore. Platforms that enable simultaneous multithreading, machines with multiple cores, and parallel computers with shared memory all offer system support for the execution of numerous independent instruction streams, sometimes known as threads.

OpenMP was defined by the OpenMP Architecture Review Board (ARB) during the latter half of the decade of the 1990s with the intention of providing a standard method for the programming of a wide variety of SMP (Symmetric Multi-Processing) architectures. OpenMP is an API that enables multi-platform shared-memory multiprocessing programming in C, C++, and FORTRAN across a wide range of platforms, instruction-set architectures, and operating systems. As a result, OpenMP is becoming increasingly popular as a means of accelerating computation because it offers a high-level approach. Because multi-core CPUs and accelerators are becoming increasingly popular, conventional programming languages have started to incorporate built-in capabilities that may assist or enable compilers in their efforts to parallelize code, and it includes FORTRAN's DC (Stulajter et al., 2021). One contribution of this PhD work is the implementation and investigation of the performance of the OpenMP with DC within the NEM code.

Due to rising energy demand and to be economically competitive, it is better to operate nuclear reactors with longer cycles and a higher capacity for power generation. Therefore, it is necessary to develop new fuel designs that involve, for example, fuel rods with higher enrichment levels. These new designs, in turn, can alter some of the safety criteria/restrictions that have been established for nuclear power reactors' operation. Consequently, safety systems as well as safety-related parameters must be analyzed in detail to ensure, under the worst possible circumstances, that the reactor will remain in a safe condition.

During the last few decades, we have seen significant developments in computing hardware and software and associated technology. The primary advancements are related to the processing speed of computer programs and the amount of memory available for storing data, i.e., computation efficiency. This fact had a significant influence on NPPs simulations, as they employ extensive numerical simulations, allowing for the multi-physics coupling of codes for more accurate safety analysis calculations.

A multi-physics coupling is performed using state-of-the-art codes for each of the physics that are important for this type of analysis. The sub-channel code CTF (Salko et al., n.d.) and its fuel rod solver CTFFuel (Toptan et al., n.d.) are utilized in for thermal-hydraulic modeling and simulation (M&S). For neutronics, the North Carolina State University (NCSU) Reactor Dynamics and Fuel Modeling Group (RDFMG) neutron diffusion code NEM is used. The coupling is done using a partitioned approach known as PI (Picard Iteration) method, which was found to be susceptible to the over-solving drawback (i.e., each PI coupled step involves excessive convergence of the constituent individual codes).

An improvement by adjusting the PI method to increase its efficiency (i.e., overcoming the over-solving drawback) by using the Residual Balance Method (RBM) resulting in better acceleration of the convergence rate of the coupled-codes system is seen as another contribution of this work. Finally, this improved coupled-codes system will be used to simulate nuclear reactor cores with high-fidelity including pin-wise and assembly-wise scales of spatial resolution models. Comparison against the PARCS/CTF (Abarca et al., 2011) coupled code system is also done to verify the developed coupled code system (NEM/CTF/CTFFuel).

The integration of the coupled codes is carried out with the assistance of MPI (Message Passing Information), which is accountable for the transfer of the necessary information between

the various codes or modules (Groop et al., n.d.) . It was designed to make portable programming easier for distributed-memory architectures, which are computer systems in which numerous processes run in parallel and communicate with one another by passing messages containing data as it is required. As a result, the codes continue to keep their initial capabilities, which means that they are able to continue operating independently of one another. The effectiveness of this integration process is also evaluated in this thesis.

Consequently, the following is a summary of the contributions of the PhD thesis:

- Acceleration of the NEM code by improving CMR.
- Implementation of the OpenMP to improve the acceleration of the NEM code.
- Implementation of the RBM to increase the efficiency of the PI.
- Integration of the couped code through the assistance of MPI.

1.2 Thesis Outline and Layout

This thesis is arranged in six chapters.

Chapter 1 contains the thesis objectives and contributions, as well as the outline of the whole dissertation.

Chapter 2 includes the literature review of previous work that has a relation to the thesis. This chapter is divided in four sections. The first section provides an introduction of the literature of the multi-physics coupling system. The second section provides a literature review of the multi-physics coupling methods. The third section provides a literature review of the acceleration methods for coupled codes and the fourth section provides the literature review of methods used to accelerate the neutronics code.

Chapter 3 is composed of three sections presenting the theory development. The first section presents an introduction. The second section presents the acceleration methods applied in

Nodal Expansion Methods. The third section presents the residual balance Method improvements. The fourth section presents the summary of the coupled convergence methods.

Chapter 4 is composed of two sections. The first one describes the temporal coupling, and the second section describes the spatial coupling.

Chapter 5 presents the results and discussions, and it is composed by three sections. The first section is focused on the development of code verification and validation. The second section presents the analysis of the results comparing to TMI Benchmark and PARCS code, for standalone case and coupled case, as well as the performance of the CMR and OpenMP for assembly wise geometry models. The third section presents the analysis of the results comparing to C5G7-TD Benchmark and PARCS code, for standalone case and coupled case, as well as the performance of the CMR and OpenMP for pin wise geometry models.

Chapter 6 presents the conclusions and potential future work.

CHAPTER 2: LITERATURE REVIEW

2.1 Multi-physics Codes System

One of the primary goals of the multi-physics coupling is to improve the predictive capability of the single-physic simulation tools used for safety analysis of current Light Water Reactors (LWRs) by dynamically coupling the codes that simulate the different physics involved in the problem into a common multi-physics simulation scheme.

The coupling between the Monte Carlo Code MCNP5 and the system thermal-hydraulics code RELAP5 were performed for determining power and temperature distributions in a VHTGR (Very High Temperature Gas Reactor) core (Conlin et al., 2005). Given that doing these computations in a VHTGR would require a cross-section library for each material at each temperature, the authors created a method for reducing the number of cross-section libraries necessary for each nuclide by employing "pseudo-materials" in which two cross-section libraries of the same nuclide at different temperatures are mixed to generate a material at the desired temperature.

The Monte Carlo method capability to perform nuclear core simulations were demonstrated with thermal-hydraulics feedback from a Computational Fluid Dynamics (CFD) code in reference (Seker et al., 2007). The platform named as McStar was written to couple MCNP5 to the CFD code STAR-CD. The library of cross-sections was created using temperature distributions at the sub-pin level calculated by STAR-CD, and the MCNP input files were handled to reflect the moderator density distribution provided by STAR-CD. The coupled codes were tested using a single PWR (Pressurized Water Reactor) pin-cell problem and 3x3 array of PWR fuel pins.

The MCNP5 Monte Carlo code and the computational fluid dynamic code, STAR-CCM+ (Neil, n.d.) were coupled to provide high fidelity results of coolant temperature and density, fuel

temperature and power distribution. Another coupling between the Monte Carlo neutronics code SERPENT and the CFD code STAR-CCM+ (Bennett et al., n.d.) were developed to model a spacer grid for a BWR (Boiling Water Reactor) core. The codes were externally coupled using I/O and the calculation was performed using the fixed-point iteration.

Although the coupling of the different Monte Carlo codes to thermal-hydraulics codes or to CFD was demonstrated in the literature, such couplings entitle long simulation times. This is due to the stochastic nature of the Monte Carlo method that needs a large number of particles during a simulation to obtain good results. Hence, deterministic codes coupling is usually used for multi-physics transient analysis of nuclear reactors as explored next.

A multi-physics platform including thermal-hydraulic, neutronics and thermo-mechanics tools were developed in 2017 (Abarca Giménez et al., 2017a). This platform can simulate the behavior of nuclear reactors and their actuation limitations, enabling the safety analyses for transients and fundamental accidents. The system code TRACE and the sub-channel code CTF, were used for thermal-hydraulic modeling and simulation (M&S). The PARCS three-dimensional nuclear reactor core simulator code was used for neutronics M&S. The FRAPCON/FRAPTRAN computer tools were used to analyze the behavior of nuclear fuel rods. These codes were integrated into the same computer platform, allowing them to communicate with one another inside the platform during simulations. The semi-implicit coupling between the thermal-hydraulic codes TRACE and CTF, an internal coupling between CTF and PARCS code and an explicit coupling between CTF and FRAPTRAN were implemented. The parallelization and optimization were performed using the MPI.

CTF, DeCART (deterministic core analysis code) and BISON (nuclear fuel performance code) codes were coupled externally (Davis et al., n.d.) with the purpose of develop an accurate

way to predict the distribution of hydrogen in nuclear fuel cladding. CTF provided fuel temperatures, moderator temperatures and moderator densities to DeCART to improve the cross-section generation. As feedback, DeCART provided pin-wise axial and radial power distributions. The convergence between CTF and DeCART provided better boundary conditions to BISON calculations.

The codes DYN3D and ATHLET were coupled (Kozmenkov et al., 2015) for modeling steady-state and transient behavior of LWR's. Internal and external coupling schemes were used to integrate these codes. Within external coupling all physical phenomena were modeled by DYN3D, while ATHLET simulated the rest of reactor system. In the internal coupling, the thermal-hydraulic was calculated by ATHLET and the neutron kinetic was calculated by DYN3D. In parallel mode, both the ATHLET and the DYN3D thermal-hydraulic core M&S ran in parallel in the same iteration loop.

An external coupling between the Monte Carlo Code SERPENT and the sub-channel code SUBCHANFLOW (Gaston, Newman, et al., 2009) was developed to serve as reference for deterministic reactor dynamics. The development was verified by code-to-code comparison with the deterministic DYNSUB coupled code.

Another multi-physics platform named MOOSE (Multiphysics Object-Oriented Simulation Environment) was developed by the Idaho National Laboratory. MOOSE began in 2008, and it is a parallel computation framework developed to solve all systems in a fully coupled way (Gaston, Hansen, et al., 2009). MOOSE solves physical problems utilizing a collection of fully coupled nonlinear partial differential equations and the JFNK approach as a parallel nonlinear solution. In comparison to traditional codes, the framework of MOOSE-based applications implies a quality paradigm where applications can be coupled tightly over various space and time scales

to achieve a multiscale approach called “MOOSE MultiApps and Transfers”. This multiscale technique enables the execution of both tightly coupled and loosely coupled models

In 2010, the US Department of Energy (DOE) established the Consortium for Advanced Simulation of Light Water Reactors (CASL) (*The Consortium For Advanced Simulation Of Light Water Reactors_ Http://Www.Casl.Gov* , n.d.) to enhance nuclear energy's economic competitiveness and safety. Integrated and complex multi-physics coupling methodologies were developed to address multi-physics difficulties in nuclear systems, such as radiation transport, thermal-hydraulics, fuel performance, and corrosion chemistry. CASL has formed a unique collaboration with four DOE national labs, three colleges, and three businesses. The consortium includes Oak Ridge National Laboratory, INL, Los Alamos National Laboratory, and Sandia National Laboratory, as well as three industrial organizations (Electric Power Research Institute, Tennessee Valley Authority, and Westinghouse Electric Corporation) to apply existing modeling and simulation capabilities and develop advanced capabilities to create an environment for predictive simulation of LWR’s. This environment is called Virtual Environment for Reactor Applications (VERA). VERA brings together capabilities to handle specific nuclear concerns and provides an extendable software architecture for developing and maintaining analytical applications and tools for the next many decades. VERA consists of four main components: physics components, numerical tools for solving linked-physics issues, drivers that execute the individual and coupled components, and infrastructure for developing high-quality software in a collaborative setting. The multi-physics coupling in VERA is based on high-fidelity coupling between the neutron transport code MPACT, thermal-hydraulics sub-channel code CTF, and the nuclear fuel solver CTFFuel.

A coupling between the Monte Carlo Code MCNP5 and the sub-channel code SUBCHANFLOW was implemented (A. Ivanov et al., 2013), to improve the precision and degree of spatial and energy resolution of core design studies. The feedback between the thermal-hydraulic and the neutronics codes is the focal point of this coupling, which considers the efficiency of Monte Carlo solutions. To the automatic transfer of the data between the codes, were developed PERL-scripts. To read the axial power profile were created a neutron mesh tally and MCNP5 tally, where the mesh tally was chosen according to the axial nodalization of the sub-channel code. The coupling scheme was tested in BWR and PWR 3x3 pin cluster problem and HCPWR (High Conversion PWR) fuel assembly. The convergence was achieved with a fair amount of iteration loops, and the analysis revealed that the coupling technique may be improved.

The hybrid system MCNP/CTF/NEM/NJOY were implemented using a methodology for generation and interpolation of the thermal-dependent thermal scattering cross section library for MCNP5 (Puente Espel, 2010). The thermal–hydraulic feedback modeling to Monte Carlo core calculations, creation and interpolation methods for temperature dependent cross-section libraries, and acceleration strategies for Monte Carlo based coupled calculations are the aims of this coupled technology. It was decided to employ combined MCNP5/CTF calculations to compute the 3D fission source distributions, as well as thermal and hydraulic parameters, to increase the accuracy of the communicated data. The nodal diffusion method NEM was utilized to generate the core configuration and fission source distribution for the MCNP5 calculations to save computation time.

2.2 Multi-physics Coupling Methods

Numerical simulations are important for forecasting the different physical processes that occur in a nuclear reactor. Through such simulation, the reliability of nuclear reactors designs as

well as their safety can be assessed. In this section some multi-physics coupling methods are summarized.

2.2.1 External and Internal Coupling

Traditionally coupled neutron kinetics/thermal-hydraulic (NK/TH) tools for reactor analysis rely on exceptionally efficient single-physics programs that have been carefully explored and verified, and which are coupled together in such a way that the result of one code becomes the input of the other code. This technique is described by the Operator Splitting (OS) methodology. It implies as a result that the impact of the governing equations on the variables has been partitioned into distinct, uncoupled physical descriptions for each component, resulting in inconsistent handling of nonlinear characteristics. By changing the solution from each physical field at each time step in the OS approaches, the algorithms are explicitly related to one another. The great advantage, that the codes are already developed and validated, can be used and slight modifications are required. In particular, the explicit treatment of the coupling variables has the effect of causing nonlinear terms to be treated inconsistently. This is the fundamental downside of the OS coupling. This coupling is referred to as "weak", since the output of one code becomes the input of the other without achieving convergence, inaccuracy is introduced. This technique is showed in Figure 2-1 (Wang et al., 2020).

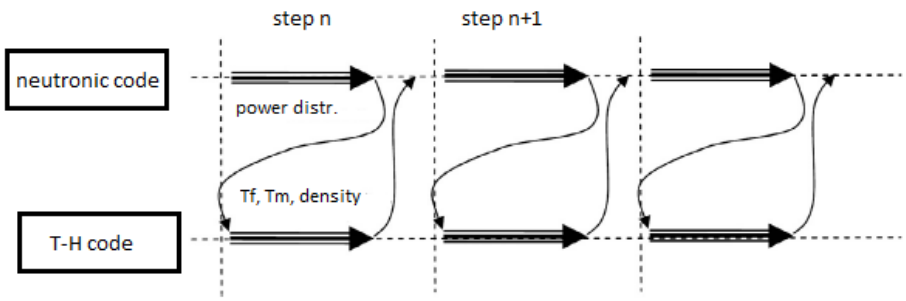


Figure 2-1: General Operator Splitting Scheme.

2.2.2 Jacobian-Free Newton-Krylov Methods

When applied to nuclear reactors, the OS technique places restrictions on numerical instability and precision. To overcome this limitation, the JFNK (Jacobian-Free Newton-Krylov) method is utilized, which is a close approximation of the well-known Newton method. The JFNK method is a fully implicit technique to solve non-linear equations (Wang et al., 2020). When solving a set of nonlinear equations, this approach utilizes the Newton and Krylov methods to ensure that the solution is both efficient and correct. The benefit of the JFNK approach is that it just requires the Jacobian matrix-vector product, rather than requiring the solution of the Jacobian matrix itself. Each time step consists of three parts: external Newton iteration, internal Krylov iteration and preconditioning.

Preconditioning increases the efficiency of the JFNK approach by lowering the number of iterations required to approximate the Jacobian matrix's inverse. The selection of an appropriate preconditioning is critical since it assists in reducing the amount of data stored in Krylov vectors and the amount of processing time required.

When employing the JFNK approach for coupling, the residuals of all the variables of the connected sets of partial differential equations (PDEs) should be created and recorded concurrently at each step of the Newton iterations, and all the equations should be reassembled into a single set of PDEs. Existing programs must be significantly modified if they are to be coupled using the JFNK technique unless the solvers are already based on JFNK methods. As a result, another coupling mechanism known as Approximation Block Newton (ABN) approaches is supplied for the sake of modularity and convenience of implementation (Wang et al., 2020).

2.2.3 Approximate Block Newton Methods

Each time Newton iterations are done with the JFNK method for coupling, all the residuals of all the variables in the PDEs should be made and recorded at the same time. Then, all the equations should be reassembled into one single set of PDEs. As pointed out before, to use the JFNK method to connect existing codes, you'll need to make a lot of changes to the codes unless the solvers already use JFNK methods. For this reason, a new approach to coupling, known as ABN, has been developed (Wang et al., 2020). Different coupling solutions may be regarded black-box solvers in this strategy, which is a substantial benefit to inherit from the research and validation effort on these existing solvers. After years of research, several derivations and descriptions of ABN techniques have been provided, but each ABN approach is based on the same strategy: the coupling issue is reformulated as a Newton solution procedure using just iterations of the original solvers, namely fixed-point iteration.

2.2.4 Picard Iteration Method

Another method used to solve multi-physics problems is denominated PI method. The PI is the most simple and robust approach for solving systems of nonlinear equations by successive approximations, i.e., it is an iterative method in which numerical solutions get increasingly accurate as the procedure is repeated. For multi-physics applications the PI method could be considered as an extension of the OS approach. Convergence in PI is almost always linear, which can lead to delayed convergence and, as a result, poor performance. Over-solving is another factor contributing to the bad results. Over-solving is a well-known phenomenon also in single-physics solvers, and it has been investigated in the context of the Newton method. The approach calculates the next update step for the outer loop by solving an inner linear iteration. Every iteration brings the inner problem closer to a solution with great precision. Given that the solution evolves because

of the feedback received from the other solver, it is not required to get 100% accuracy in every iteration of the algorithm. To control local convergence and, as a result, over-solving in Newton solvers, inexact Newton techniques employing various methodologies are presented (DEMBOt et al., n.d.). Figure 2-2 (Senecal & Ji, 2017a) shows the PI for coupling of two solvers.

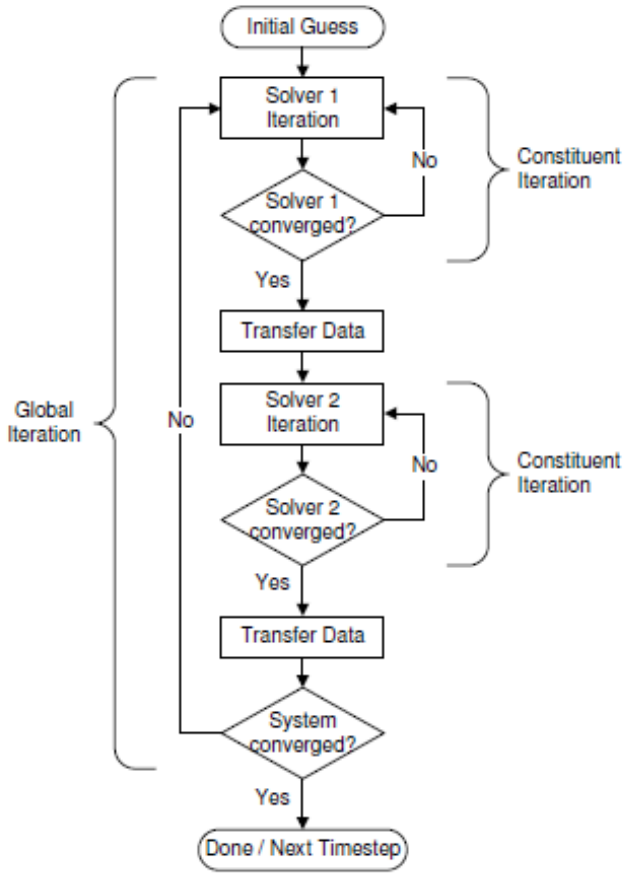


Figure 2-2: Description of the PI Using Two Solvers.

2.3 Acceleration Methods for Coupled Codes

Each solver in PI receives input from the others, computes a new solution, and then sends that data to the others after it has been updated. Because each single-physics problem is tackled individually and repeatedly until the final answer, performance deteriorates. The poor performance is due to two factors: linear convergence rate and over-solving. Hence, it is of importance to resolve

such drawbacks of the widely used PI method. Numerous strategies have been developed to accelerate the rate of convergence. Different acceleration techniques, including Richard extrapolation, Aitken delta-squared process, and Wynn epsilon method, have been proposed (*C. Brezinski - Convergence Acceleration during the 20th Century*, n.d.). Anderson acceleration (*A. R. Toth - A Theoretical Analysis of Anderson Accelertion and Its Application in Multiphysics Simulation for Light-Water Reactors*, n.d.) is another approach that has been developed and studied for the goal of speeding the convergence rate in coupled situations.

The partitioned multi-physics solver technique has been used to generate a variety of coupling approaches that have been implemented inside the computational framework. Changes to the commonly used PI approach led to the creation of several more efficient algorithms (Senecal & Ji, 2017a)(Senecal & Ji, n.d.).

The modifications are predicated on the general concept of reducing the termination need in order to improve computation efficiency. When there are a lot of different single-physics solvers, both the convergence behavior of each one and the strength of the connection between them affect how well it works. In weakly coupled problems, the feedback between the single-physics solvers can be solved with a small number of global iterations. As the coupling gets stronger, it takes more global iterations to solve for feedbacks effects, which means the computational costs go up. When solving reactor challenges that have a high degree of connection between neutronic and thermal-hydraulic systems, imprecise techniques should be used to avoid excessive repetitions. These approaches developed by Senecal are discussed below.

2.3.1 Solution Interruption

In this technique, the constituent solvers must pause their own solution iteration process before achieving complete convergence and then send data to the next solver. This is done by

forcing one solver to perform only one constituent iteration and then transferring the result to the other solver, which completes the solution. Each global iteration resolves one problem completely and another partially. This process is presented in Figure 2-3 (Senecal & Ji, 2017b). Due to the asymmetrical nature of the Solution Interruption approach, extreme caution is required when selecting the inner application. Because the Solution Interruption approach reduces the amount of work carried out by the outer solver, it is typically desirable to select the less expensive application as the inner solver. This allows one to take advantage of the fact that the Solution Interruption method. When compared to using an outer solver, using an inner solver can result in significant savings in terms of computational time and effort.

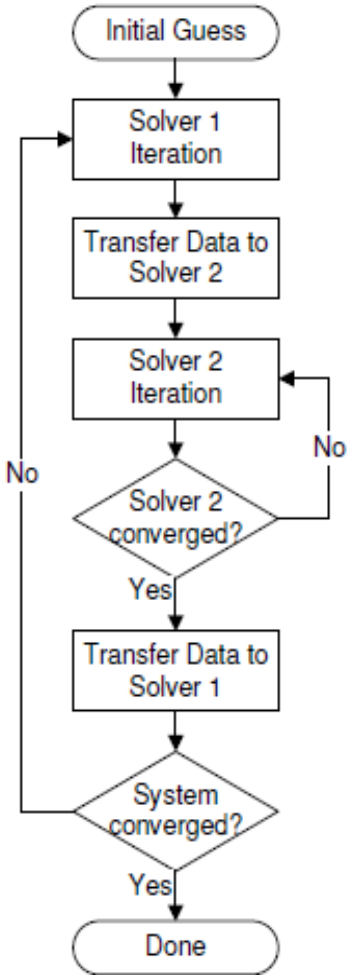


Figure 2-3: Solution Interruption Method.

Another advantage of adopting the Solution Interruption approach is that it may fit an inner program that is a black box. This is because the inner application does not require any interruptions or modifications. Since the inner solver is always entirely solved, the coupled simulation only needs to run until the outer solver converges. The inner solver is always converged, so its status doesn't need to be checked when the coupled problem's convergence is being checked.

2.3.2 Alternating Nonlinear Method

At each global iteration, each solver in the alternating nonlinear technique performs only one iteration step. This technique minimizes constituent iterations at the price of increased global iterations. Because the Alternating Nonlinear technique requires more global iterations than the PI method for weakly coupled problems, the time savings achieved by the Alternating Nonlinear method will rely on the relative cost savings of global vs. component iterations. Figure 2-4 (Senecal & Ji, n.d.) presents the flow chart of the alternating nonlinear method.

2.3.3 Relaxed Relative Tolerance

Senecal modified the solver settings of component applications to reduce over-solving in PI. In other words, the easiest way to stop over-solving might be to change the solver parameters for each application. In a normal setting, each application will normally have a relative tolerance parameter that requires full convergence on each iteration at the global level. Over-solving is decreased by simply lowering this setting because full solves are substituted with partial solves. Birken's work (Birken, 2014) developed a nonstandard convergence criterion to reduce the over-solving. In a further example, Lipnikov (Lipnikov et al., 2013) utilized an adaptive inexact solution method in order to handle nonlinear advection diffusion problems within the setting of finite volumes. In both cases, the relative tolerances of the constituent solvers are loosened up in order to facilitate a more rapid acceleration of the global convergence.

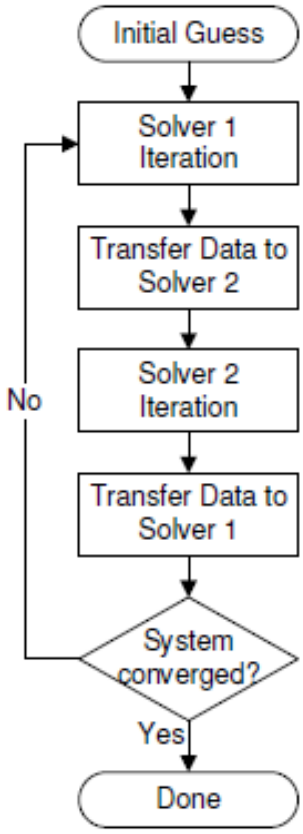


Figure 2-4: Alternating Nonlinear Method.

Since the coupled system is not solved until the given tolerance on the overall residual is fulfilled, it is quite acceptable in practice to reduce the residual of a constituent solver by, say, only one decade per global iteration. As a result, there will be no reduction in the precision of the linked solution. It is the form of Picard Iteration that a lot of experts recommend using since avoiding over-solving with it is simple and straightforward, which is why it is the recommended option. The flow charge of this method is equal to PI (Figure 2-2), except that the relative tolerance for each solver is relaxed.

2.3.4 Residual Balance

The convergence criteria of the constituent solvers can be relaxed based on the norm of their residuals with respect to each other using the Residual Balance Method (RBM), which is a variation on PI. When one single-physics solver's residual norm drops below the residual norm of another single-physics solver, the latter solver takes over and begins the process of finding a solution. This way, the solvers continue changing the information based on the comparison of the norm of their residuals.

Two changes must be made in order to use it in a multi-physics code for PI: the absolute tolerance must be adaptive, and each solution must perform at least one constituent iteration when called. The procedure is similar to PI, except that the solver convergence criterion has been changed. The initial L2 norm of the residual is stored in memory by the first solver in the RBM before the iteration of a single component is finished. The L2 norm of the residual is then computed and stored by the second solver. The second solver iterates through k ($k \geq 1$) constituent iterations until one of the following requirements is fulfilled (Senecal & Ji, 2017a):

$$\left\| R_k^{S2} \right\| < abs\_\text{tol} \tag{2.1}$$

or in terms of relative tolerance

$$\frac{\|R_k^{S2}\|}{\|R_0^{S2}\|} < rel\_tol \tag{2.2}$$

The absolute tolerance is given by: $abs\_tol = max(a\ \|R_0^{S1}\|, \min\ \_abs\_tol)$ and the value of $a = 0.1$ (Birken, 2014) was demonstrated to be a good value with the best CPU time. Because the RBM shares information about the state of the solution with all of the solvers, it makes it possible for intelligent adjustments to be made to the tolerances of the solvers. The RBM and the improvements it brings will be explored in detail in Chapter 3.

2.4 Standalone NEM Acceleration Methods

The standard inner/outer iteration multigroup diffusion theory approach is used to solve the multigroup Nodal Expansion Method numerically. That is, inner iterations or repeated sweeps of the mesh with a known internal source is done for each group to invert the diffusion removal matrix inside the group. Following the inner iterations, outside or fission source iterations are done to determine the right values of the problem's multiplicative eigenvalue and the spatial and energy dependent fission neutron source distribution. To satisfy the convergence conditions on the effective neutron multiplication factor keff and the fission source vector, the NEM solution approach inherently needs an excessive number of outer iterations. Another situation (Chao, n.d.-a) where many outer iterations are needed is when steep slopes occur at assembly interfaces, e.g., (Bandini, n.d.-b) on the assembly interfaces in Mixed Oxide (MOX) cores where absorption/fission cross-sections are different between different assemblies. Hence, acceleration methods are needed to accelerate NEM outer and inner iterations.

Several techniques for acceleration can be found in literature for NEM equations. This includes Coarse Mesh Finite Difference (CMFD) (Chao, n.d.-b), Coarse Mesh Rebalancing (CMR), and Multi-Level Coarse Mesh Rebalancing (ML-CMR) (Bandini, n.d.-a), Wielandt-shift method (Bandini, n.d.-b), parallelization, to name a few.

2.4.1 Coarse Mesh Finite Difference (CMFD) Method

The CMFD method has been used to effectively accelerate diffusion and neutron transport calculation. The CMFD method uses coupling coefficients to construct a very basic nodal balancing equation with just node average fluxes as unknowns, and it became a practice to solve nodal methods with the purpose to accelerate the calculation. CMFD correction coefficients are added and repeatedly updated across two neighboring nodes utilizing the emerging advanced nodal

technique solution. The correction coefficients are determined to reproduce nodal surface currents and nodal surface fluxes. Since it employs a variation of the finite difference formulation, the nonlinear acceleration technique is often referred to as the CMFD acceleration method. Due to the large reduction in memory storage requirements and calculation time associated with CMFD acceleration, it is commonly used in a variety of sophisticated nodal algorithms. The CMFD approach is particularly well-suited for big systems, such as the whole core geometry of commercial reactors.

2.4.2 CMR with Wielandt-shift and Asymptotic Extrapolation

Throughout the process of working through a variety of steady-state sample issues, it was found that the fundamental NEM solution approach invariably calls for an excessive number of outer iterations. Following an analysis of several different iterative acceleration strategies, we determined that the most effective strategy for accelerating the basic NEM would be to use coarse-mesh rebalancing in conjunction with asymptotic extrapolation and Wielandt-shift method (Bandini, n.d.-b).

The components of the coarse mesh consist of a reduction in the projecting of the multigroup NEM diffusion equation towards a coarse-group diffusion-like equation with lower dimensionality (e.g., one coarse group). The amount of computational effort required to carry out numerous iterations of a coarse mesh is therefore almost nothing when compared to the cost of doing a single whole-core multigroup diffusion sweep. This is because of the simplified model that still catches the main numerical trends (e.g., eigenvalues and dominance ratio) of the original solved matrix.

Although rebalancing provides the primary convergence acceleration effect for both steady-state and transient computations, occasional asymptotic extrapolations can still, under

certain conditions, provide significantly more than merely a second-order additional acceleration effect. Because asymptotic extrapolations can more closely approximate the behavior of an asymptotically convergent function, the asymptotic extrapolation is then utilized as a supplementary acceleration mechanism for computations that take significantly more time to complete transient.

When applied in NEM computation (Bandini, n.d.-b), asymptotic extrapolation resulted in a reduction in the number of outer iterations that is comparable to a factor of two. As a result, the combination of coarse mesh rebalancing and asymptotic extrapolation, which is applied to most realistic situations, reduces the number of outer repetitions, improving the Nodal Expansion Method's computing efficiency.

During the solution of the CMR, as explained in Section 3.2.1, the new system of equations generated uses the power iteration method to find the associated eigenvalue. It is then possible to accelerate the power iteration method by using the Wielandt fractional iteration method (or more commonly known as the shift method). In this method, the shifted eigenvalue problem is solved instead of the original eigenvalue problem. The idea is to shift the original k-eff eigenvalue (the largest eigenvalue) by a known amount (e.g., 1.05) such that the dominance ratio (that is, the ratio between the second largest and the largest eigenvalues) is smaller resulting in a faster convergence for the power iteration method.

These three methods, when combined show good acceleration for both assembly-wise and pin-wise problems and are included in NEM. Modifications to these methods for increasing stability and increasing the acceleration outcome will be explained in both Chapters 3 and 5, respectively.

2.4.3 Multilevel Coarse Mesh Rebalance

A methodology for solving differential equations by employing a hierarchy of discretization is referred to as the MG method. This method is used in numerical analysis. They are an example of a class of approaches known as multiresolution methods, which are extremely helpful in solving issues that display behavior on numerous scales. The main idea of multigrid is to accelerate the convergence of a basic iterative approach by a global correction of the fine grid solution approximation from time to time, performed by addressing a coarse issue. The multi-level rebalancing makes use of several different coarse mesh layers.

Framatome's whole-core neutronics and thermal hydraulics core simulation program ARTEMIS implemented the multi-level critical boron iteration method. This multi-level technique is distinguished by the projection of variational boron concentration changes to coarser mesh levels inside a multi-level rebalancing hierarchy connected with the nodal flux equations to be solved in steady-state core simulation (van Geemert, n.d.). The optimized variational criticality tuning algorithms were implemented at each individual rebalance mesh level. This method was employed by default in ARTEMIS reactor cycle simulations, and all predictions of the critical boron concentration for all burnup time steps were more accurate than 0.001 ppm in a computer-friendly manner. This level of precision is essential for the precise optimization of industrial core simulations and the proper evaluation of reactivity perturbations.

2.4.4 Parallelization and System based Optimizations

High-performance computing necessitates the utilization of numerous cores and parallel processing, but the software implementation can be approached in a variety of different ways. Most of the historical code that needs to be ported to more recent systems (current boards that incorporate powerful, multiprocessors) is serial code. This means that the code operates on a single

processor, and there is only one instruction that executes at a time, which is the case for the NEM code.

To update a serial program so that it can be run in parallel, you must first determine which individual pieces of the program can run at the same time, and then you must optimize those sections so that they can run on many cores and/or processors at the same time (*Tammy Carter - Choosing a Parallelization Technique*, n.d.). In addition, parallel programs need to make use of a control mechanism to achieve coordination, synchronization, and data realignment. The process of parallelization can be made easier with the assistance of a wide variety of open standard tools, which can be found in the form of language extensions, compiler extensions, and library extensions (Kiessling, 2009). The primary rationale for parallelizing your code is to reduce the amount of time it takes to run. Such tools (e.g., gprof (Blaise Barney, n.d.), MAQAO (https://pop-coe.eu/sites/default/files/pop_files/pop-webinar-maqao.pdf, n.d.), Intel VTune (https://hpc.llnl.gov/software/development-environment-software/intel-vtune-amplifier, n.d.)) will be used to profile the NEM code and the results are provided in Chapter 3.

Parallel programming became popular in the early 1990s when physical sciences and engineering needed larger and faster answers than individual computers could handle. What used to be an impossible-to-solve problem became possible because each computer could work on a small part of the problem and only share the information that other computers needed to move forward with their calculations. This led to message-passing parallel programming, which became the MPI library. MPI then became the standard for parallel FORTRAN, C, and C++ programming (Curcic & Rouson, n.d.) for distributed memory systems (i.e., multi-computational nodes clusters).

Abarca (Abarca Giménez et al., 2017b) created a computational platform that allowed thermal-hydraulic, neutronics, and thermos-mechanical codes automatically interact with one

another. The system code TRACE and the sub-channel code COBRA-TF are used when dealing with thermal-hydraulic issues. The PARCS three-dimensional diffusion code is utilized when dealing with neutronics. The fuel performance codes FRAPCON and FRAPTRAN are utilized too. The fact that all these tools are included within the same computer platform makes it possible for them to communicate with one another while the simulations are being run. The integration of the various coupled codes is accomplished through the utilization of an MPI parallel execution library. This library is tasked with the responsibility of transferring the necessary information between the various codes or modules. As a result, the codes continue to keep their initial capabilities, which is to say that they can continue operating independently.

Coarrays were added to the 2008 Standard to offer an efficient and robust parallel model. Coarrays are comparable to arrays and employ the same indexing method. The Coarray programming model provides a simple syntactic modification for labor and data distribution in parallel programming (Metcalf et al., n.d.). The Coarray programming model consists of two new features that have been added to the language. One of these features is an extension of the normal array syntax that represents data decomposition. The other feature is an extension of the execution model that controls the distribution of parallel work. The Coarray execution concept is based on Single-Program-Multiple-Data (SPMD) concept, where program replications operate independently in separate memory areas (Padua, n.d.).

A parallel process, often known as a thread or a core, is what is meant by the term "Fortran image." Each image has its own independent copy of the software and its own memory that is local to it. The program is executed by each picture separately from the others unless it is expressly commanded to do it differently. The process of telling the images to wait for one another is referred

to as "synchronizing the images," and it is feasible to issue such a command (Curcic & Rouson, n.d.).

Another parallelization capable system that has gained importance during the 2000s with the introduction of multicore processors, is the shared memory systems. Both OpenMP and OpenACC are common directive-based (Application Programming Interface) that parallelize code on shared memory systems so that it can operate on multi-core CPUs and GPUs respectively.

The United States Nuclear Regulatory Commission (US NRC) is responsible for developing the Purdue Advanced Reactor Core Simulator (PARCS) code, which is used for the safety study of nuclear reactors that are operational in the United States. The time-dependent Boltzmann transport equation necessary to calculate the neutron flux distribution in the core of the nuclear reactor is solved by the code. The code was created in FORTRAN, and it was tested on a wide range of computer platforms and operating systems before it was released. On multiprocessor workstations from SUN and SGI, the nuclear reactor transient analysis code PARCS was modified to make use of POSIX Threads and OpenMP to provide parallelism (Lee D.J. & Downar T.J., n.d.) since Pthreads does not allow an interface to FORTRAN. There was work done on a threads library that can be used by both FORTRAN and C code. It is made up of a small set of thread functions for FORTRAN that were implemented by "wrappers" that went from FORTRAN to C. One way to make a program work with multiple threads is to "fork" each subroutine that has a parallel section. The benefit of this method is that you don't have to worry as much about global vs. local variables. But it could slow down performance because the function that creates the Pthreads has extra work to do. This is especially true if the subroutine runs repeatedly. It's possible that the extra work that comes with that function could cancel out any performance boost from doing things in parallel. To avoid this problem and make the application to PARCS as efficient as possible, the

main program was split at the start of execution. OpenMP is a standard set of programming instructions that can be used with FORTRAN, C, and C++ on a computer with a shared address space. Since OpenMP is based on directives, it is much easier to implement than Pthreads. The parallelism in the OpenMP version of PARCS (like domain decomposition) was almost the same as parallelism in the Pthreads version. Using Pthreads and OpenMP, two parallel versions of the PARCS nuclear reactor transient analysis code were made. There are still some problems with how Pthreads and OpenMP are used in SGI and SUN. But overall, OpenMP on SGI performed about the same as Pthreads on SUN. The performance was different in each module because of how the calculations were done. Based on cache hit times, a simple predictive model was made, and the results were close to the measured performance. For parallel programming on a shared memory address machine, the directive-based OpenMP standard seems to be the best choice in terms of how hard it is to implement.close to the measured performance. For parallel programming on a shared memory address machine, the directive-based OpenMP standard seems to be the best choice in terms of how hard it is to implement.

Miko M. Stulajter et al (Stulajter et al., 2021) investigated the present capabilities, portability, and performance of replacing directives with FORTRAN's do concurrent using a mini app that currently implements OpenACC for GPU acceleration and OpenMP for multi-core CPU parallelism. Multiple settings and compiler options (GCC's gfortran, NVIDIA's nvfortran, and Intel's ifort) were examined, and several directives were changed to read "Do Concurrent (DC)". The purpose is to test replacing directives with DC for accelerated computing, guaranteeing that multi-core CPU parallelism is preserved, and that performance is comparable to that of the original directive-based code. DC is a new variation of the do construct that enables parallel execution of loop iterations to increase efficiency in FORTRAN. It was possible to remove and replace all

directives and obtain efficient CPU and GPU parallelism using the nvfortran. This, however, required specific nvfortran capabilities, such as implicitly recognizing reductions and the usage of unified managed memory.

2.5 Conclusions and Summary

Calculations of the neutronics/thermal-hydraulic coupling are of particular importance because they have the potential to make a major contribution to both the enhancement of the reactor design and the satisfaction of the ever-increasing requirements for reactor safety. Improving coupling approaches in terms of the accuracy and efficiency of numerical simulations is the primary problem in the field of neutronics/thermal-hydraulic calculations (K. Ivanov & Avramova, 2007). "Best-estimate" modeling for nuclear system analysis necessitates an expanded usage of coupled three-dimensional neutron-kinetics and thermal-hydraulic system codes to anticipate realistic safety margins.

It is evident that multi-physics problems are more difficult and complex than problems that can be solved using models of a single physics because it is necessary to understand not only the law governing a single physics for each phenomenon, but also the interplay between different physical processes. This is because it is necessary to understand not only the law governing a single physics for each phenomenon (Senecal J. P., n.d.).

There are numerous coupling methodologies that can be employed in multi-physics coupling; however, the Picard Iteration is one type of coupling that is often used in simulations of this type. This strategy makes it possible to directly exploit current and well-developed single-physics systems without having to rewrite substantial chunks of the code. Single-physics codes iteratively pass solutions to each other as inputs in Picard Iteration until each code has reached a converged solution. However, multi-physics computing coupled by Picard Iteration is prone to

over-solving, which can make the whole computation much less efficient than it otherwise would be.

There are many methods to solve the over-solving problem in multi-physics coupling were introduced. Special attention will be given to Solution Interruption Method, and Residual Balance Method. These methods are a variation on the Picard Iteration coupling method, and they are used in the single-physics codes that include adaptive termination criteria that are inaccurate (Senecal & Ji, 2017c).

Three numerical strategies, known as coarse-mesh rebalancing, asymptotic extrapolation and Wielandt fractional iteration method that improve the convergence of the outer iteration in nodal expansion methods and they will be explored in this thesis (Bandini, n.d.-b).

CHAPTER 3: THEORY DEVELOPMENT

3.1 Introduction

Because of its simplicity, flexibility, and ease of implementation and application, the PI method is widely considered as the most straightforward and robust strategy for solving nonlinear systems of equations.

As referred before, this sort of iteration works by solving single-physics applications sequentially and transmitting the changed state variables or response functions to the other applications, iterating in this fashion until a consistent solution or some measure of failure is achieved.

As a result of the input from one solution, the residual norm of the other solver may rebound to a higher level. An inexact method which modifies the termination criteria of the constituent solvers should be employed to minimize these unnecessary iterations. Moreover, acceleration of individual codes in a coupled-codes system should also help with the global convergence rate of the PI method. Both topics are explored in the next sections.

In this regard, and from the different acceleration techniques reviewed in the literature in Chapter 2, we will focus on the methods implemented or improved in the codes used in this thesis.

3.2 Acceleration/Optimization Methods in NEM

The NEM code has been used in 30+ theses since it was developed and introduced in the 1990s. Several researchers and graduate students have worked with it in their thesis work where they implemented new ideas and physics inside this nodal expansion method reactor simulator code. Without a modern build system (e.g., CMake) or a version control system (e.g., Git), the different previous implementations and additions to the code were not tabulated and was not easy to follow.

One of the first steps done in the NEM code then was to optimize and rewrite portions of the code to follow modern programming paradigms (e.g., loop fusion, object orientation, and helpful comments) which would help with the code being concise and easy to work with. Furthermore, GitLab (an internet-based software development and version control system using Git) was introduced as a measure for version control of NEM to facilitate working with the code further which allowed accelerated development of the new ideas in NEM discussed in the next sections.

Since different libraries were going to be used in NEM for coupling or for parallelization (e.g., MPI and OpenMP) as well as for solving systems of linear equations (e.g., LAPACK), a modern build system had to be adopted for the code. CMake build generator was used in this regard and a dedicated shell script to compile the code and to use CMake to include different build options was made. The script is labelled 'install_NEM.sh' and can be used directly in the root directory of the NEM installation. This script's help document is shown in Figure 3-1. To invoke this document, one can just execute the script with the '--help' or the '-h' options which would show the results of Figure 3-1, which also shows the capabilities of compiling the NEM code.

```
[mraltahh@rdfmg nem]$ ./install_NEM.sh -h

Welcome to the NEM installation script.

The default options used by the install script:
Type:               release
Compiler:           gnu
Format:             new
Parallel library:   serial
Math library:       default
Cluster type:       rdfmg
Compiler edition:   old


Syntax: install_NEM [-h|t|c|f|p|m|l|e]
Default options are printed in bold green color.
options:
-h, --help       Prints this help and exits.
-t, --type       Assigns the build type.           From the following arguments:  release , debug, test.
-c, --compiler   Assigns the compiler.             From the following arguments:  gnu , intel, nvidia.
-f, --format     Assigns the input format.         From the following arguments:  new , native.
-p, --parallel   Assigns the parallelization type. From the following arguments:  serial , mpi, openmp, openacc, openmp+mpi.
-m, --math       Assigns the math library.         From the following arguments:  default , mkl, user.
-l, --cluster    Assigns the cluster type.         From the following arguments:  rdfmg , user.
-e, --edition    Assigns the compiler edition.     From the following arguments:  old , new.
```

Figure 3-1: Help Document for Installing the NEM Code Invoked by "./install_NEM.sh -h" in the NEM Root Directory.

Such work (code revision, optimizing, and using a build system) also could result in acceleration. This acceleration not only affects the code computations (as is shown in Section 3.2.1 due to the different code optimizations carried out), but it also indirectly affects how fast one can implement the new methods explored in this thesis (e.g., parallelization). This last part comes by allowing a concise way to deal with the code compilation (e.g., when having both MPI and OpenMP loaded together for the coupled systems) and version control.

In the next sections, we will explore the new acceleration techniques added to the NEM code, as well as the enhancement done for the previous acceleration techniques found already in NEM. Details and effects of the code acceleration on both standalone and coupled simulations are adjourned to Chapter 5.

3.2.1 CMR Method with Wielandt-shift and Asymptotic Extrapolation

The NEM solution methodology inevitably requires an excessive number of outer iterations to meet the convergence criteria on keff and the fission source vector. Methods to speed up or

accelerate the convergence of the outer fission source iterations can mitigate this issue (Bandini, n.d.-b).

The coarse-mesh rebalancing technique works by computing a system eigenvalue and a series of neutron flux/partial current scale factors. When applied, these scale factors will force a neutron balance over a set of coarse mesh cells, each of which is arbitrarily made up of one or more of the originally defined fine mesh cells. The coarse-mesh rebalancing method can be derived by defining the overall neutron balance equation for an arbitrary coarse node m:

$$-\sum_{l\in S_m}\Delta^2(x,y,z)\sum_{g=1}^{2}J_{g(x,y,z)}^{in,l}+\sum_{l\in S_m}\Delta^2(x,y,z)\sum_{g=1}^{2}J_{g(x,y,z)}^{out,l}+\sum_{m\in V_m}\Delta x\,\Delta y\,\Delta z\sum_{g=1}^{2}\sum_{ag}^{m}\bar{\phi}_g^m=\frac{1}{k_{eff}}\sum_{m\,\epsilon\,V_m}\Delta x\,\Delta y\,\Delta z\sum_{g=1}^{2}\nu\sum_{fg}^{m}\bar{\phi}_g^m \quad (3.1)$$

The previous equation describes neutrons disappearing from node m through absorption, fission, and outgoing currents, while incoming neutron currents from adjacent nodes describes the amount incoming. This equation will not be exactly satisfied if the outer iterations have not converged. The balance can be artificially satisfied, however, by defining a set of rebalance factors. The coarse mesh fluxes and outgoing partial currents are modified as follows:

$$\bar{\phi}_g'^m \equiv f^m\,\bar{\phi}_g^m \quad (3.2)$$

$$J_{g(x,y,z)}'^{out,m} \equiv f^m\,J_{g(x,y,z)}^{out,m} \quad (3.3)$$

The terms on the right-hand side (RHS) are the pre-balanced terms (obtained from preceding NEM outer iteration), while the ones on the left-hand side (LHS) are the re-balanced terms (obtained from coarse rebalancing, used for the subsequent NEM outer iteration).

Substituting for the rebalanced terms in the coarse nodal balance equation, we will obtain an equation that can be solved for the rebalancing factor f^m. These rebalance factors are calculated from the following eigenvalue equation:

$$Mf=\frac{1}{\lambda}Pf \quad (3.4)$$

where M is a matrix containing the combined absorption and currents terms, while P is a diagonal matrix containing the fission term and λ is the eigenvalue of this equation, and f is the vector of rebalance factors for each node inside the system. The calculated rebalance factors will all tend towards 1 as the outer iterations converge.

To solve this eigenvalue equation, the Wielandt method of fractional iteration can be applied to the power iteration method for the numerical solution of eigenvalue problems. The so-called shifted eigenvalue problem is formulated first:

$$M' = \frac{1}{\lambda'} Pf \tag{3.5}$$

where:

$$M' = M - \frac{1}{\lambda_e} P, \quad \frac{1}{\lambda'} = \frac{1}{\lambda} - \frac{1}{\lambda_e} \tag{3.6}$$

Here λ_e is an estimate of k_{eff} for the problem fundamental mode eigenvalue. In this analysis λ_e is defined such that:

$\lambda_e = 1.05 * k_{eff}^{n-1}$ is the estimate of k_{eff} from the previous outer iteration. The idea by applying this method is that the dominance ratio of Equation (3.5) is smaller than that of Equation (3.4) and hence would be faster to converge as explained in Section 2.4.2. The power iteration method is employed in the solution of Equation (3.6) by using the following sequence:

$$S^t = \frac{(M'_{currents} \times S^{t-1} + P \times f^{t-1})}{M'_{absorption}} \tag{3.8}$$

$$\lambda' = \frac{P \times S^t}{P \times f^{t-1}} \tag{3.9}$$

$$f^t = \frac{S^t}{\lambda'} \tag{3.10}$$

where t is the power iteration index, $M'_{currents}$ is the portion of the M' matrix that includes the currents only, while $M'_{absorption}$ is the portion of the M' matrix that includes the absorption

term. Equation (3.8) requires an inner iteration for the S vector components to converge which is tested by using point norm of the S vector components between successive iterations. Equations (3.9-3.10) are computed following the convergence of Equation (3.8), completing the shifted power method iteration.

After the convergence of the power iteration method, the calculated value of f^m is multiplied to all nodal unknowns in node m, and then the rebalanced terms are used as the initial conditions for next NEM outer iteration.

CMR then provides a suppression of otherwise slowly decaying low-frequency non-fundamental mode components in the not-yet-converged diffusion iteration. This is realized through the systematic multiplicative correction of nodal fluxes and interface currents, prior to each full core NEM diffusion sweep, with iteratively obtained ratios between coarse mesh re-balanced and pre-balanced terms.

In NEM, asymptotic extrapolation is used as an auxiliary acceleration mechanism with the CMR method to improve the convergence of the outer iterations. In cases where the convergence is still slow, the asymptotic extrapolation can supply additional convergence (Bandini, n.d.-a).

This straightforward extrapolation approach is based on the premise that the fission source vector, $\bar{S}^n_{total}$, converges asymptotically to its fully converged solution, $\bar{S}^\infty_{total}$:

$$\bar{S}^\infty_{total} = \bar{S}^n_{total} + \vec{R}\,\sigma^n \tag{3.11}$$

$$\sigma^n \cong \frac{\left\|\bar{S}^n_{total} - \bar{S}^{n-1}_{total}\right\|_2}{\left\|\bar{S}^{n-1}_{total} - \bar{S}^{n-2}_{total}\right\|_2} \tag{3.12}$$

where σ^n is the dominance ratio of the matrix that yields the flux solution and $\vec{R}$ is an unknown vector. Combining the two previous equations, the estimate of the fission source vector is determined as:

$$\bar{S}'^n_{total} = \bar{S}^n_{total} + \omega^n\,(\bar{S}^n_{total} - \bar{S}^{n-1}_{total}) \tag{3.13}$$

$$\omega^n \equiv \frac{\sigma^n}{1-\sigma^n} \tag{3.14}$$

This extrapolation is performed when the following criteria is satisfied:

$$\varepsilon_\omega^n < 0.1 \tag{3.15}$$

$$\varepsilon_\omega^n = \left|\frac{\omega^n - \omega^{n-1}}{\omega^n}\right| \tag{3.16}$$

The σ^n is applied to the iteration n node flux moments and outgoing partial vectors before the next iteration n+1, according to:

$$\bar{\phi}_g'^{l(n)} = \bar{\phi}_g^{l(n)} + \omega^n\left(\bar{\phi}_g^{l(n)} - \bar{\phi}_g^{l(n-1)}\right) \tag{3.17}$$

$$J_g'^{out,l(n)} = J_g^{out,l(n)} + \omega^n\left(J_g^{out,l(n)} - J_g^{out,l(n-1)}\right) \tag{3.18}$$

When applied, the asymptotic extrapolation in conjunction with the coarse-mesh rebalancing reduces the number of outer iterations thus improving the computational efficiency of the nodal expansion method.

3.2.2 NEM Acceleration Methods Stabilization and Optimization

The previously described methods manage to accelerate the computations significantly when applied during the outer iterations of the NEM equations. However, it was discovered that for some numerical experiments (e.g., for multigroup problems) or for some configurations (e.g., CMR with asymptotic extrapolation), the methods could lead to divergence resulting in not harvesting the acceleration benefits of the three methods working together. Hence, it was required to investigate and to modify the code to increase the computations speed while having a stable code.

First, for the asymptotic extrapolation method, Equation (3.15) was changed to consider the relative change from the previous extrapolation iteration:

$$\varepsilon_\omega^n,\ \varepsilon_\omega^{n-1} < 0.1 \tag{3.19}$$

Here one defines the convergence as being asymptotic if the relative change (cf. Equation (3.16)) in the extrapolation parameter ω^n at the present and previous iterations are less than 10% from their previous iterations respectively. This modification helps with stabilization of the extrapolation method and once added, it allowed to use the extrapolation method with the CMR method for all tested numerical experiments.

On the CMR side, two modifications were made to both accelerate and stabilize the method. Starting with the stabilization modification first, it should be noted that in NEM computations, we use both removal (that is, absorption and out-scattering from some energy group g) and absorption cross-sections. Unfortunately, the NEM code computed the absorption cross-sections from the removal cross-sections by subtracting from it the out-scattering cross-sections contribution. For multigroup computations or for benchmarks that require reading the cross-section from dedicated tables, the CMR method always diverged. This was corrected by using different variables for each cross-section and thus stabilizing the associated computations significantly (e.g., using CMR for full-core acceleration while using the fine mesh).

The acceleration of the CMR was carried out by using two methods. The first method was to realize that Equation (3.8) is an iterative method and hence it would be better to always use the previous iteration as the onset value for the current iteration. Previously, this value was always initialized to be zero between successive iterations (i.e., $S^{t-1} = 0$ at the onset of a power-method iteration). Removing this term (thus making the code use the previous value as the current value $S^t = S^{t-1}$ for the current power-method iteration) helped with accelerating the convergence by around two folds as discussed in Section 5.3. The second acceleration is realized by applying parallelization to the different parts of the CMR methodology, which is explained and explored in Section 3.2.4.

3.2.2.1 Profiling of the NEM code

The parallelization of the NEM code was carried out after analyzing the code using different profiling tools. Profiling is a software engineering and development concept in which a profiling tool (e.g., gprof) analyze a computer code dynamically (e.g., while executing the code) to measure the space (i.e., memory) usage or time taken for different portions of the code, as well as the frequency of procedures call. Apparently, such profiling helps with identifying hotspots inside a computer code (that is, locations inside the code that have adverse effects on the code execution).

Profiling is mostly done by compiling the computer code using specific compilers flags (i.e., profiling by instrumentation) that help another tool called the profiler to carry out profiling-analysis of the code using different techniques like event-based or statistical methods (i.e., sampling). The two main profilers that were used for this purpose in the thesis are gprof and MAQAO. Intel VTune was also considered, however since the RDFMG cluster (that is, the computers cluster on which NEM/CTF-CTFFuel compilation and execution are done) is based on AMD processors computational nodes, it would not have been an option since intel codes (e.g., VTune or Inspector) run only on intel processors.

3.2.2.2 gprof Sampling Profiler

gprof is a performance analysis tool that comes with the GNU project set of programs, and hence it can be found on many Unix/Linux systems as the RDFMG cluster. During the compilation process, an instrumentation code is inserted into the program code gathering caller-procedure data (by using a compilation flag. E.g., in the GCC set of compilers, this is done by using the “pg” flag). The sampled data is saved in an output file just before the code finishes execution and this data

can be analyzed with gprof command-line tool. An output of the gprof program for running the NEM code is shown in Figure 3.2.

```
Flat profile:

Each sample counts as 0.01 seconds.
 %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
61.47      7.21     7.21        1     7.21    11.40  __nem_steady_MOD_solve_ss
33.84     11.18     3.97       64     0.06     0.06  __nem_iter_MOD_inner_cartesian
 1.19     11.32     0.14   220576     0.00     0.00  __nem_tab_MOD_interp_xs
 0.60     11.39     0.07   220576     0.00     0.00  __linear_interpolation_module_MOD_initialize_2d
 0.60     11.46     0.07       25     0.00     0.00  __nem_acceleration_MOD_cmr_coarse_to_fine_mesh
 0.51     11.52     0.06       26     0.00     0.00  __nem_acceleration_MOD_coarsemeshrebalance
 0.51     11.58     0.06        1     0.06     0.07  __nem_in_MOD_read_nemin
 0.43     11.63     0.05       32     0.00     0.00  __nem_acceleration_MOD_store_old_leakage_moments_and_flux
```

Figure 3-2: Example of gprof's Flat Profile Result Output File Generated by Running the NEM Code on the TMI Benchmark Problem.

Figure 3-2 shows gprof's output and the profile of the code. This profile has the execution time of each procedure (as well as call counts) and its percentage of the total running time. The output is sorted by percentage showing the hot spots at the top of the generated list. Two points can be seen in these figures. First, although the reading of the cross-sections' tables and the associated interpolation procedure is carried out more than 200,000 times, but the time taken is less than 0.15 of a second. On the other hand, the steady state solver inner iteration subroutine has been called 64 times and it amounts to one-third of the total time spent by the code.

The large number of calls for the interpolation subroutine is justified by the TMI benchmark data. 438 different materials cross-sections and 11 state points (per material) cross-section are read for them to be interpolated to the problem current state point. Since the interpolation process used in NEM is linear, this quadrature like computations does not result in a bottleneck. This is especially since the compilation flags used also include unrolling and vectorization of the code loops making it easy for the processor to process such large number of simple arithmetic iterations.

The specific 64 calls of the inner iteration subroutine can be explained by the fact that the outer iterations taken in this NEM code run was 32. And since we have 2 energy groups in the TMI benchmark, this then justifies the 64 calls of the inner iteration procedure. However, such 64 calls need to be optimized (i.e., the inner iteration procedure) since the code takes more than one-third of its total time inside this subroutine.

To see more detailed profiling, one should resort to an advanced profiling tool like MAQAO or intel's VTune. Since VTune usage on the Linux cluster was not an option, MAQAO was used for more detailed profiling of the current code.

3.2.2.3 MAQAO

MAQAO is a suite of tools for profiling, analyzing, and optimizing HPC applications. It provides a low-overhead profiler capable of handling any runtime, LProf; a quick, yet accurate, static analyzer, CQA, which allow to tune your code for the very specific characteristics of your micro-architecture; and a result aggregator module, OneView, presenting performance reports and pinpointing bottlenecks.

The tool mixes both dynamic and static analyses based on its ability to reconstruct high level structures such as functions and loops from an application binary. Since MAQAO operates at binary level, it does not require recompiling the application to perform analyses. MAQAO assesses the code quality of the most time-consuming loops and provides a best-case estimation of the performance that can be reached, along with some hints on how to achieve it in terms of source code transformations, compiler flags, pragmas, etc.

To generate a detailed profiling of the NEM code, MAQAO was applied to the NEM executable while using the TMI benchmark standalone case. The executable was compiled with debug information included (i.e., with the "-g" compiler flag), such that MAQAO can show the

content of the analyzed loop. An html report is generated by MAQAO that one can use to see the code contents and how much time is taken for each function (for Fortran, procedures and modules) and loop. Figure 3-3 shows the loop profile inside the NEM program. This figure shows that around 28 inner loops (that is, single loops that do not contain any kind of loop inside) covers more than 8% of the program runtime. Hence if optimization is to be done, it would be mostly with these inner loops such that they cover less time from the program runtime.

By inspecting the functions and loops analyzing tabs in MAQAO's report, we find that the inner iteration module takes around 66% of the total time spent by the NEM code. Of these 66%, loop number 611 takes around 16% of the program runtime alone. This loop can be inspected by MAQAO as seen in Figure 3-4. Loop 611 is basically a reduction loop (i.e., it reduces the array dimension of some variable) which can be parallelized using OpenMP directives. Furthermore, MAQAO can give an expert summary of the loops that are considered as hotspots and require optimization as seen in Figure 3-5. It is noteworthy that one can click on any loop ID to see the loop contents, or one can resort to the source code with the specific lines provided by MAQAO in the last column of the summary in Figure 3-5.

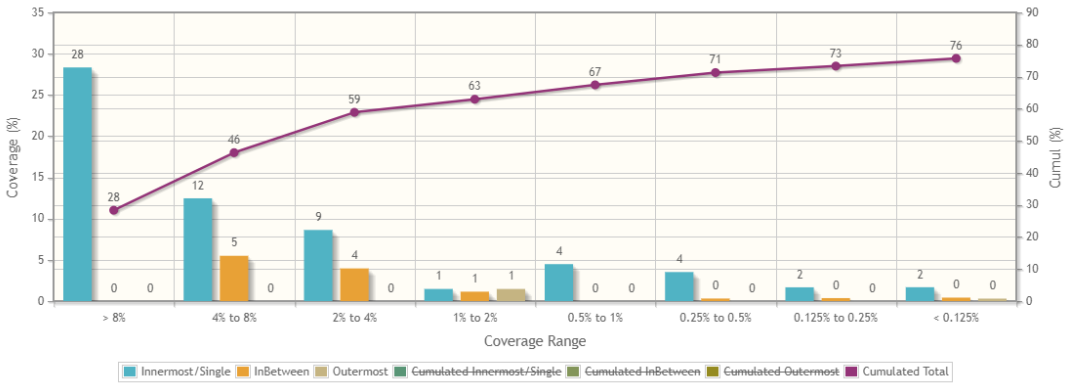


Figure 3-3: Loop-Based Profile of the NEM Program.

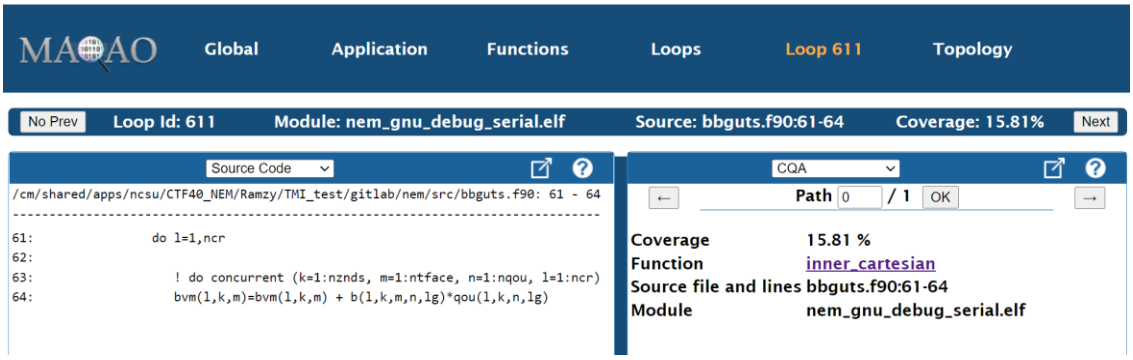
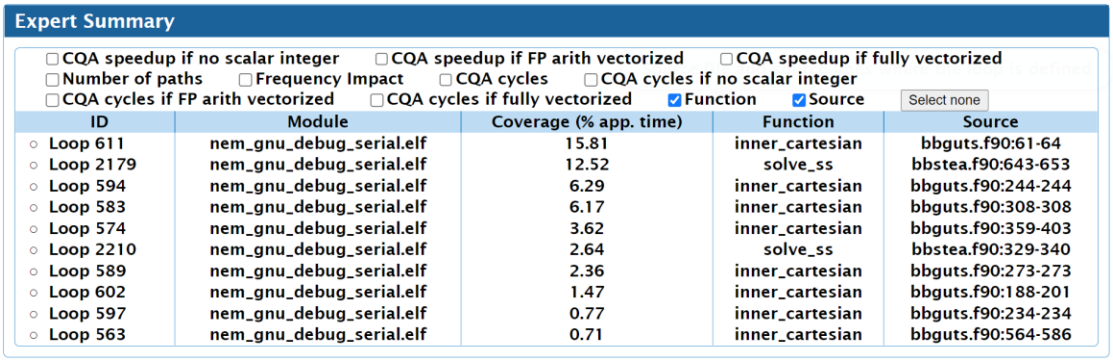


Figure 3-4: Loop-611 Time Coverage of the Total Wall Time and its contents.



ID	Module	Coverage (% app. time)	Function	Source
Loop 611	nem_gnu_debug_serial.elf	15.81	inner_cartesian	bbguts.f90:61-64
Loop 2179	nem_gnu_debug_serial.elf	12.52	solve_ss	bbstea.f90:643-653
Loop 594	nem_gnu_debug_serial.elf	6.29	inner_cartesian	bbguts.f90:244-244
Loop 583	nem_gnu_debug_serial.elf	6.17	inner_cartesian	bbguts.f90:308-308
Loop 574	nem_gnu_debug_serial.elf	3.62	inner_cartesian	bbguts.f90:359-403
Loop 2210	nem_gnu_debug_serial.elf	2.64	solve_ss	bbstea.f90:329-340
Loop 589	nem_gnu_debug_serial.elf	2.36	inner_cartesian	bbguts.f90:273-273
Loop 602	nem_gnu_debug_serial.elf	1.47	inner_cartesian	bbguts.f90:188-201
Loop 597	nem_gnu_debug_serial.elf	0.77	inner_cartesian	bbguts.f90:234-234
Loop 563	nem_gnu_debug_serial.elf	0.71	inner_cartesian	bbguts.f90:564-586

Figure 3-5: Expert Summary of the Loops that Require to be Revised and Considered for Optimization.

MAQAO can also provide hints for improving performance as well as hotspots/bottlenecks description. However, such an ability needs MAQAO to have the executing processor registered in its internal list of processors (which mostly contains intel manufactured processors). Since the RDFMG cluster's AMD processors used in this analysis were not supported by MAQAO, such capability was not leveraged.

Profiling of the NEM code proved to be important. It showed the loops inside the code that require optimization and reconsideration. Some of these loops can benefit from parallelization (e.g., loop-611 reduction) and some should require reformatting (e.g., changing the array

multiplication form to make it easier for the compiler to optimize). In the next section, we will give a brief introduction to the parallelization technique and apply it to the NEM code, giving some examples from the code in the process.

3.2.3 Techniques of Parallelization

The need for ever-increasing computational capability led computer system designers to consider combining many of their existing computing devices. This is where parallel computing began, opening a whole new world for researchers and programmers (Hermanns, 2002).

3.2.3.1 Parallelization Process

The development of codes that can use the capabilities of the existing hardware to tackle larger problems in a shorter amount of time is one of the challenges presented by parallel computers. It is possible to distinguish primarily two families of parallel machines:

Shared-memory architecture: these parallel machines are constructed using a set of processors, each of which can access a shared memory space. Computers that are built on this design are typically referred to as SMP machines (Figure 3-6) (Curcic & Rouson, n.d.). In a shared-memory system, multiple CPUs share memory (RAM).

Distributed-memory architecture: each CPU in these parallel machines has access to its own private memory, and information is passed between the processors by way of messages. This category of computer hardware is frequently referred to as clusters (Figure 3-6) (Curcic & Rouson, n.d.). In a distributed-memory system, each CPU has its own memory and exchanges data via a network, represented by the dashed lines. Commonly, the distributed-memory system consists of multicore shared-memory systems.

Parallel computers with shared memory, multicore processors, and simultaneous multithreading platforms all permit the execution of multiple independent instruction streams, or

threads (Kiessling, 2009). By combining these technologies, it is possible to build machines with many threads they can run. These technologies can be put together to make computers that can run a lot of threads (a thread is a single sequential flow of control within a program) at once.

Loop-level parallelism is a type of software programming parallelism focused on the extraction of parallel jobs from loops. In computer applications that store data in random-access data structures, loop-level parallelism is frequently possible. Whereas a sequential program iterates through the data structure and operates on indices one at a time. A program using loop-level parallelism uses several threads or processes that operate on some or all the indices simultaneously. This parallelism accelerates the overall program execution time. In parallel computing, Amdahl's law is frequently used to predict the theoretical speedup when using additional processors. This law also represents the fact that the overall performance improvement gained by optimizing a single part of a system (e.g., a Do-loop) is limited by the fraction of time that the improved part is used.

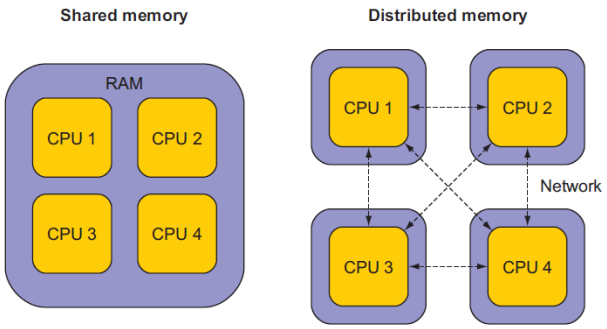


Figure 3-6: Shared Memory and Distributed Memory.

Software has traditionally been written for serial computation Figure 3-7 (Blaise Barney, n.d.):

- A problem is divided into discrete series of instructions
- Instructions are executed sequentially one after the other

- Executed on a single processor
- Only one instruction may be executed

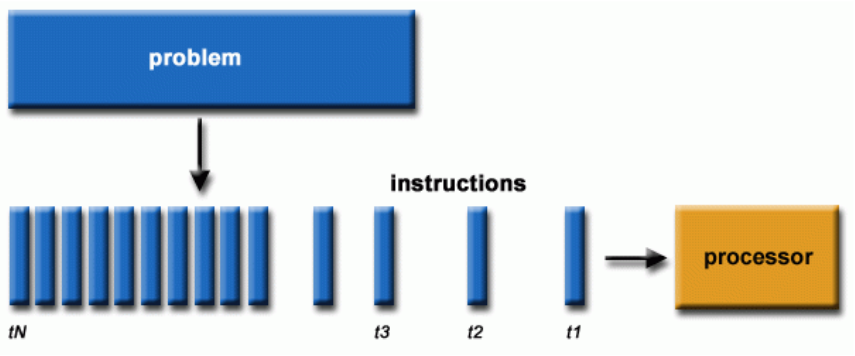


Figure 3-7: Serial Computing Example.

Parallel computing is the use of many compute resources to solve a computational issue at the same time Figure 3-8 (Blaise Barney, n.d.):

- A problem is divided into distinct components that can be resolved simultaneously.
- Each part is further broken down into a set of instructions.
- Instructions from each section run concurrently on various processors.
- A global control/coordination mechanism is used.

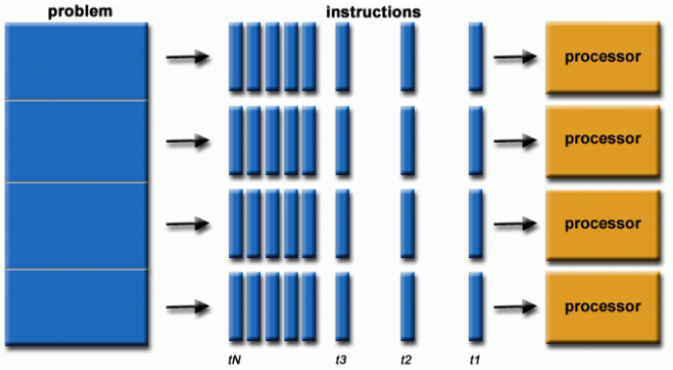


Figure 3-8: Parallel Computing Example.

Through the technique of splitting a problem into multiple tasks and performing them simultaneously on a multicore or multiprocessor system based on shared or distributed memory architectures, parallel computing can tackle issues that need a significant amount of computational power. The new era of multicore computing requires software programmers to design parallel programs to make full use of the parallelism offered by the hardware. In order to automate the process of transforming sequential code into parallel code, tools for automatic parallelization have been developed (Prema & Jehadeesan, 2013). Techniques for parallelization, such as Dependency Analysis and Loop Parallelization, play a significant part in the overall process of parallelization.

3.2.3.2 Message Passing Information

The Message Passing Interface, also known as MPI, was initially developed in the early 1990s by a consortium of researchers and vendors who based their work on the APIs that were already being used by those vendors (Chapman Barbara et al., n.d.). In high-end computing, where tasks are typically so complex that they require the combined efforts of many machines, MPI is a prevalent programming paradigm.

As a result of the relative ease with which it may be implemented across a large number of different platforms, it offers high portability. However, there is a cost associated with its portability. Using this application programming interface (API) to create a parallel program often needs a significant rearrangement of the sequential code that was originally used. When compared to a technique that is supported by the compiler, such as that which is provided by OpenMP, the programming effort can be large and difficult. It is also possible to mix some programming APIs. In particular, MPI and OpenMP can be used together in a program, which can be helpful if the program is going to be performed on MPPs that are comprised of numerous SMPs (possibly with multiple cores each) (Chapman Barbara et al., n.d.). Utilizing the hierarchical parallelism that is inherently present in either the application or the underlying hardware can be accomplished through the combination of MPI and OpenMP.

The hybrid programming approach achieves its highest level of productivity when MPI processes operate at a coarse level of parallelism and make use of the user-controlled data distribution and job scheduling. For extra fine-grained parallelization, the shared address space of each process is utilized with parallelization that is based on OpenMP (Chapman Barbara et al., n.d.).

Marco Aldinucci et al (Aldinucci et al., 2021) proposed a semi-automatic methodology for parallelizing scientific applications developed with a purely sequential programming perspective. They show that the same parallelization method works in the shared memory model (using OpenMP), the message passing model (using MPI), and the general-purpose computing on GPU model (via OpenACC).

3.2.3.3 Open Accelerators

OpenACC is a programming model that was introduced in 2011. It makes use of high-level compiler directives to expose parallelism in the code and parallelizing compilers to create the code for a range of parallel accelerators (*OpenACC Programming and Best Practices Guide*, n.d.). Since OpenACC is a directive-based model that enhances a given code base with clues, the code can be simply compiled in a serial manner, ignoring the directives, and still provide accurate results. This is possible since OpenACC is a model that augments a given code base with hints. As a result, it is feasible to preserve or maintain a single code base while also delivering portability across more than one platform (Chandrasekaran S. & Juckeland G., 2018). OpenACC is not a language. It consists of compiler directives, library functions, and environment variables.

In order to annotate a program's source code, OpenACC makes use of something called directives. These directives are interpreted by a compiler that is capable of OpenACC, which then builds parallel executable code according to the directives' specifications. If the code is compiled using a compiler that is not capable of OpenACC or with OpenACC interpretation disabled, the compiler will ignore the directives and produce a serial program that does not contain any parallelism (Kopysov et al., 2006). After the compiler directive keyword (!$ for Fortran), the directive type (which is “acc” for OpenACC directives) comes next. The actual directive that tells the compiler what to do comes next, followed by one or more clauses that can be added if more information is needed. A directive is a piece of code that gives "instructions" to the compiler about how to process the code block that comes after it. There are three distinct sorts of directives in OpenACC (Chandrasekaran S. & Juckeland G., 2018):

1. Compute directives - denote a section of code in which you can increase performance by using the inherent data parallelism and distributing work across numerous threads.

Parallel, *kernels*, *routine*, and *loop* are the compute directives that are supported by OpenACC.

2. Data management directives - OpenACC programs should avoid needless memory data movement and using only computer directives can cause data movement because the compiler must be conservative and assure a serial state. Data, update, cache, atomic, declare, enter data, and exit data are OpenACC data directives.
3. Synchronization directives - OpenACC enables task parallelism, allowing many constructs to run simultaneously.

3.2.3.4 Open Multi-Processing for NEM Code

OpenMP is a shared-memory application programming interface (API) whose capabilities are built on past efforts to promote shared-memory concurrent programming, as described in Figure 3-6. OpenMP is not a programming language and should not be confused with one. It is a notation that can be added to a sequential program written in FORTRAN, C, or C++ to describe how the work is to be shared among threads that will execute on different processors or cores and to order accesses to shared data as required (Chapman Barbara et al., n.d.).

Numerous factors have contributed to OpenMP's success. One is its emphasis on parallel structured programming. The compiler is responsible for determining the specifics of the parallel program, which makes OpenMP easy to utilize. Its widespread adoption is its primary benefit, allowing OpenMP applications to function on a variety of systems.

The directives of OpenMP (an OpenMP directive is a set of instructions written in a format that only OpenMP compilers can understand) allow the user to guide the compiler on which instructions to execute in parallel and how to distribute them across the threads that will execute

the code. Like the OpenACC directives, the OpenMP directives start with “!$” for Fortran, but then has the directive type “omp” instead of “acc” that is used in OpenACC.

Identifying the parallelism in a sequential program is the first step in building an OpenMP program from it. It involves locating instructions, sequences of instructions, and even huge portions of code that can be processed concurrently by many processors. This step is considered as the profiling step, which was carried out in Section 3.2.2.1. The second step in developing an OpenMP program is to represent the parallelism that has been uncovered by utilizing OpenMP. The great advantage of OpenMP is that can be applied to create a parallel program from an existing sequential code. It means directives can be added to part of the program while leaving the remainder sequential.

As stated before, OpenMP realizes a shared-memory programming model. This approach implies programs to run on several processors that share memory. Shared-memory programs are executed by several threads, which share data but may also have private data. Shared-memory approaches to parallel programming must provide, in addition to the usual set of instructions, a way to start threads, give them work, and coordinate their access to shared data.

OpenMP uses a “Fork and Join” parallel execution architecture, and this is shown in Figure 3-9 (Kiessling, 2009). The master thread is the first process that runs in any OpenMP software. The execution of this master thread is sequential until a parallel section is reached. The master thread now "forks" into a few concurrent worker threads. This team of worker threads then executes the instructions in the parallel zone. The threads synchronize and rejoin at the conclusion of the parallel region to form the single master thread once more. That is, in the fork and join model, OpenMP programs start with a master thread that runs in order until it reaches a section

that runs in parallel, at which point the master thread splits into multiple working threads. At the end of the parallel section, the threads rejoin the master thread.

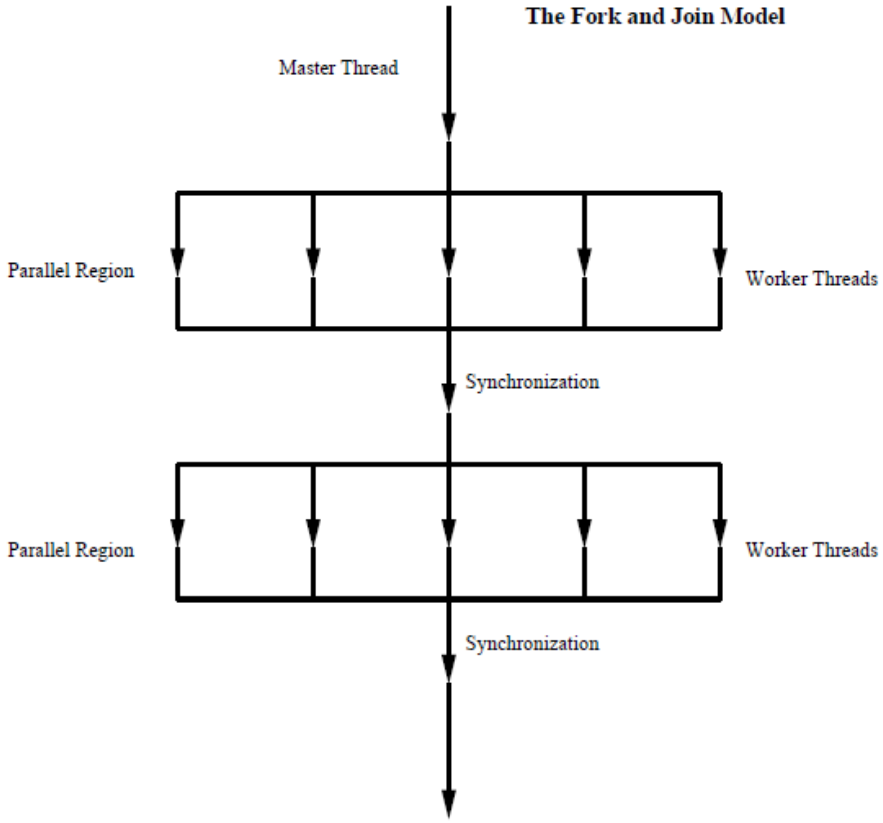


Figure 3-9: Model of Forks and Joins used in OpenMP.

Like a sequential program, OpenMP begins with a single thread of execution. The first thread to run this code is called the initial thread. When a thread runs into an OpenMP parallel construct, it does the following: It makes a group of threads while the program is running (this is the fork), becomes the leader of the team and works with the other members of the team to run the code inside the construct, which is dynamic (it is assigned the thread id 0 within the group). At the end of the construct, only the first thread, which is also called the "master of the team," continues (this is the join). Figure 3-10 explain how it works in NEM code.

```
!$omp parallel do shared(numco,iocol,mc,jin,mb1,qou,mb2,leak,ma) private(ipiv,brhs,alhs,info,l,lc)
SCHEDULE(AUTO)
      do k = 1, nznds
        do lc = 1, numco(icol, k)
          l = iocol(lc, icol, k)
          ipiv = 0
          ! "qou" contains the sources only without leakage.
          brhs(:) = matmul(mc (:, :, l, k, lg), jin (l, k, : , lg)) &
          &      + matmul(mb1(:, :, l, k, lg), qou (l, k, : , lg)) &
          &      + matmul(mb2(:, :, l, k, lg), leak(l, k, : , lg))
          alhs    = ma(:, :,l, k, lg)
#ifdef INTEL
          call gesv(alhs,brhs,ipiv,info)
#else
          call dgesv(6,1,alhs,6,ipiv,brhs,6,info)
#endif
          ! Problems in the solver
          if (info /= 0) then
            write(*,*) ' A*X=B solver has a problem '
            write(*,*) ' INFO =  ', info
            stop
          end if
          jout(l,k,:,lg)=brhs(:)
```

Figure 3-10: Example of Inner Loop Iteration that Solves the Outer Current Equation in NEM Code.

3.3 Residual Balance Method Improvements

3.3.1 Residual Balance method

The purpose of the Residual Balance approach is to avoid over-solving while maintaining a high rate of convergence. When the Residual Balance method is applied, the residual norm of one of the constituent solvers is lowered to a value that is a little bit lower than what it was at the beginning of the process for the other solver. This is done to prevent the problem from being over-solved (Senecal & Ji, 2018). Thus, each solver captures the progress made by the other with a bounded result.

Considering that each single solver has a residual norm, for a problem with two constituent solvers (S1 and S2) the residual balance termination criterion for the solvers is:

$$\left\|R_{i,j}^{S1}\right\| < \begin{cases} a\left\|R_{0,0}^{S1}\right\| & for\ i = 0 \\ a\left\|R_{i-1,0}^{S2}\right\| & for\ i > 0 \end{cases} \tag{3.12}$$

$$\left\|R_{i,j}^{S2}\right\| < a\left\|R_{i,0}^{S1}\right\| \tag{3.13}$$

where $\left\|R_{i,j}^{S1}\right\|$ and $\left\|R_{i,j}^{S2}\right\|$ are the residual norm of S1 and S2, respectively, and the index i refers to global iteration (some authors call Picard Iteration) and j refers to the constituent iteration. The value of the parameter "a" guarantees that the current solver approaches the magnitude of the residual norm of the other solver. Combined with the more traditional absolute and relative termination criteria, this termination criterion is used to determine when a process should be terminated. The solution with a larger residual norm can catch up with the other solver in time. If one solver already has a lower residual norm than the other, it will do very little work while waiting for the other.

The fact that the constituent single-physics problems may have residuals on different scales due to changes in physical units or discretization is one of the difficulties associated with multi-physics problems. This is especially true in multi-physics scenarios. If the scales of the two problems are very different from one another, the solution to the problem with the larger residual is going to be more accurate than the solution to the other problem with the smaller residual. In this case, it is necessary to apply normalization. The residual norm is controlled by the absolute and relative termination criteria in most iterative numerical solvers:

Absolute termination criterion:

$$\left\|R_{i,j}^{S1}\right\| < \varepsilon_a^{S1} \tag{3.14}$$

Relative termination criterion:

$$\left\|R_{i,j}^{S1}\right\| < \varepsilon_r^{S1} \left\|R_{i,0}^{S1}\right\| \tag{3.15}$$

The values of the tolerances ε_a^{S1} and ε_r^{S1} are set by the user.

Because alternative discretization of multi-physics problems can be used, the use of normalizing provides for improved performance in the constituent solvers.

The norm of solver S1 is normalized to:

$$\left\|\hat{R}_{i,j}^{S1}\right\| = \frac{\left\|R_{i,j}^{S1}\right\|}{\varepsilon^{S1}} \tag{3.16}$$

where ε^{S1} is the effective tolerance, and it is defined as:

$$\varepsilon^{S1} = \max\left(\varepsilon_a^{S1},\ \varepsilon_r^{S1} \left\|R_{0,0}^{S1}\right\|\right) \tag{3.17}$$

The same is done for the solver S2.

It is crucial to keep in mind that the residual balancing strategy accelerates the first stage of the solution process by performing several component iterations in the solver with a higher residual. It means, the method quickly reduces this residual component until its contribution is comparable with that of the other solver.

Therefore, the termination criterion after normalization is:

$$\left\|\hat{R}_{i,j}^{S1}\right\| < \begin{cases} a\left\|\hat{R}_{0,0}^{S1}\right\| & for\ i = 0 \\ a\left\|\hat{R}_{i-1,0}^{S2}\right\| & for\ i > 0 \end{cases} \tag{3.18}$$

$$\left\|\hat{R}_{i,j}^{S2}\right\| < a\left\|\hat{R}_{i,0}^{S1}\right\| \tag{3.19}$$

Each physics component is solved thoroughly many times until the general solution is reached in Picard iteration. When using inexact techniques (such as Residual Balance), it is impossible to ignore the contribution of the other solver to the overall residual (due to the solvers changing information). After this normalization, residuals from both solvers can be added to determine the total residual norm:

$$\|R_i^{tot}\| = \left\|\hat{R}_{i,j}^{S1}\right\| + \left\|\hat{R}_{i,j}^{S2}\right\| \tag{3.20}$$

3.3.2 Adaptive Residual Balance Method

The goal of the parameter a stated in the termination criterion is to ensure that the current solver's residual norm will be close to the magnitude of the residual norm of the other solver. The value of $a = 0.1$ used on termination criterion (defined by the user), works well for several problems (Senecal & Ji, 2017c) . Consequently, an ideal value for "a" can be established, and the employment of an automated approach to find this parameter is useful in this situation. This scheme is based on the Picard convergence rate (ρ_i). If the Picard iteration converges linearly, the effective convergence rate of the coupled issue can be determined using the ratio of the current and prior combined residual norms,

$$\rho_i = \frac{\|R_i^{tot}\|}{\|R_{i-1}^{tot}\|} \qquad for\ i > 0 \tag{3.21}$$

To determine the solver tolerance for the current global iteration, is it utilize the effective convergence rate ρ_i (relative tolerance of the system coupled). To keep the convergence rate

constant, the active solver's residual norm should be reduced by a factor of at least a = ρ_i below the other solver's initial residual norm. If ρ_i is assigned to an excessively large number, it may result in extra Picard iterations (under-solving).

The reduction factor $a = \frac{\rho_i}{2}$ is employed to maintain asymptotic convergence. Then the termination criterion for the first solver is:

$$\left\|\hat{R}_{i,j}^{S1}\right\| < \begin{cases} a \left\|\hat{R}_{0,0}^{S1}\right\| & for\ i = 0 \\ \frac{\rho_i}{2} \left\|\hat{R}_{i-1,0}^{S2}\right\| & for\ i > 0 \end{cases} \tag{3.22}$$

or in terms of normalized residuals:

$$\frac{\left\|R_{i,j}^{S1}\right\|}{\varepsilon^{S1}} < \begin{cases} a \frac{\left\|R_{0,0}^{S1}\right\|}{\varepsilon^{S1}} & for\ i = 0 \\ \frac{\rho_i}{2} \frac{\left\|R_{i-1,0}^{S2}\right\|}{\varepsilon^{S2}} & for\ i > 0 \end{cases} \tag{3.23}$$

and the termination criterion for the second solver is:

$$\frac{\left\|R_{i,j}^{S2}\right\|}{\varepsilon^{S2}} < \begin{cases} a \frac{\left\|R_{0,0}^{S1}\right\|}{\varepsilon^{S1}} & for\ i = 0 \\ \frac{\rho_i}{2} \frac{\left\|R_{i,0}^{S1}\right\|}{\varepsilon^{S2}} & for\ i > 0 \end{cases} \tag{3.24}$$

3.3.3 Coupled Convergence Methods

The technological problems that the nuclear industry encounters necessitate a high level of precision in multi-physics modeling and simulation. When using multi-physics analysis to mature and find regular use, it is critical to maintain computation costs as low as feasible. The need to implement a more efficient tightly coupled algorithm, leaded Senecal (Senecal & Ji, n.d.) to

develop the RBM. The RBM was improved through implementations of methods used to accelerate the convergence rate and was generalized to include several residuals from the same code instead of using one global residual per code. The termination criterion used in each iterative level categorizes all these strategies used to accelerate the convergence rate and is categorized in Table 1.

Table 1: Termination Criteria for Different Coupled Convergence Methods

Picard Iteration	**Residual Balance**	**Adaptive Residual Balance**
Absolute termination criterion: $\left\|R_{i,j}^{S1}\right\| < \varepsilon_a^{S1}$	$\left\|\hat{R}_{i,j}^{S1}\right\| < \begin{cases} a\left\|\hat{R}_{0,0}^{S1}\right\| & for\ i=0 \\ a\left\|\hat{R}_{i-1,0}^{S2}\right\| & for\ i>0 \end{cases}$	$\frac{\left\|R_{i,j}^{S1}\right\|}{\varepsilon^{S1}} < \begin{cases} a\frac{\left\|R_{0,0}^{S1}\right\|}{\varepsilon^{S1}} & for\ i=0 \\ \frac{\rho_i}{2}\frac{\left\|R_{i-1,0}^{S2}\right\|}{\varepsilon^{S2}} & for\ i>0 \end{cases}$
Relative termination criterion: $\left\|R_{i,j}^{S1}\right\| < \varepsilon_r^{S1}\left\|R_{i,0}^{S1}\right\|$	$\frac{\left\|R_{i,j}^{S2}\right\|}{\varepsilon^{S2}} < \begin{cases} a\frac{\left\|R_{0,0}^{S1}\right\|}{\varepsilon^{S1}} & for\ i=0 \\ \frac{\rho_i}{2}\frac{\left\|R_{i,0}^{S1}\right\|}{\varepsilon^{S2}} & for\ i>0 \end{cases}$	$\frac{\left\|R_{i,j}^{S2}\right\|}{\varepsilon^{S2}} < \begin{cases} a\frac{\left\|R_{0,0}^{S1}\right\|}{\varepsilon^{S1}} & for\ i=0 \\ \frac{\rho_i}{2}\frac{\left\|R_{i,0}^{S1}\right\|}{\varepsilon^{S2}} & for\ i>0 \end{cases}$

CHAPTER 4: CTF/NEM COUPLED CODE

4.1 Message Passing Information – Communication Interface

Although NEM/CTF coupled code is not an original contribution of this dissertation, the author wants to introduce it because of its importance in the developments here presented. The standalone codes have been improved with the different acceleration methods here proposed, like the CMR and ML-CMR implemented in NEM, and the convergence of the coupled code itself has also been accelerated with the method proposed and described in Chapter 3.

The developed coupling algorithm is based on an MPI Coupling Interface (CI) developed over a Server-Client (S-C) communication protocol. This coupling algorithm has been successfully applied in CTF/PARCS, with highly analogies with the here presented CTF/NEM coupling. MPI has been chosen as communication protocol because of the high standardization and the flexibility provided to couple parallelized simulation codes simultaneously, taken advantage of the native parallel capabilities of CTF. Furthermore, the designed loose coupling algorithm allows the external coupling of different simulation codes without the necessity of extended modification of their compilation projects. The unique pre-requisite is the incorporation of a FORTRAN source file with the CI and the compilation of the coupled codes using an MPI compiler. In this approach the analyst must decide which simulation code acts as a server, and which are the codes that are connected to that server as clients. This selection should be done using command line arguments or defining this in the input files of the coupled codes. A general sketch with the description of the S-C MPI CI is presented in Figure 4-1, where, for example in this case, CTF will be configured as the Server and NEM as the Client1.

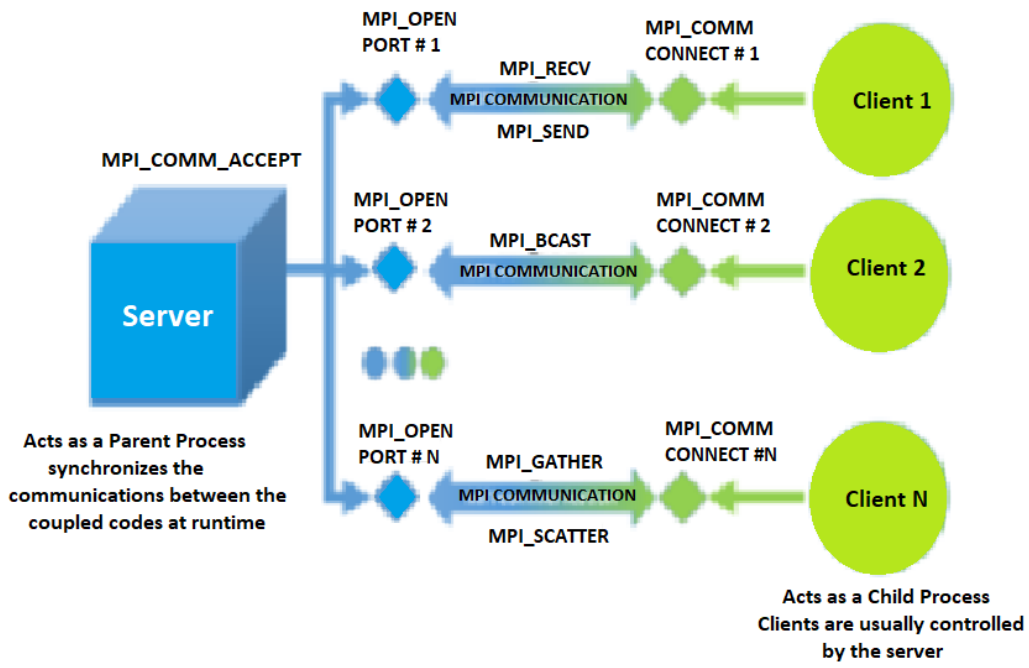


Figure 4-1: General Scheme of the Server/Client Based MPI Coupling Interface.

The main characteristics of the developed protocol have been successfully tested in two different applications as a demonstration case. The first application, development of the temporally explicit coupling between the sub-channel thermal-hydraulics code CTF and the neutron-kinetics core simulator PARCS, demonstrates the flexibility of the S-C coupling interface to be independent of the building environment and serial and parallel codes. The second application, development of a MPI version of the ECI of TRACE, shows its strength managing highly synchronized applications without a real impact on the performance.

4.2 Temporal Coupling

Temporal coupling and synchronization are highly important in the neutron-kinetics and thermal–hydraulics codes since these codes frequently employ their own time-step selection algorithms.

An external temporally explicit coupling approach is used for the coupling of CTF and NEM. When using external coupling, both programs retain their own separate executables, which increases their updating flexibility. External coupling also allows for the interchange of feedback parameters using external buffers and mapping tables. While one code is doing a computation, the other code is waiting for the computation to be completed, which justifies the use of the acceleration techniques mentioned in Chapter 3. This is known as briefly explicit coupling in programming. A master/slave relationship is used to govern the linked-codes system, with CTF serving as the master (i.e., server) and NEM serving as the slave (i.e., client). CTF sends NEM tagged messages, which are used to guide NEM activities. These tags are associated with several instructions, including initializing NEM (e.g., reading the input file by NEM), time-step advancing (e.g., in the temporal solution) or simulation finalizing which is used when a solution is found, when the simulation time runs out without convergence, or when an error occurs. Such tagged messages are also involved while exchanging feedback variables between the two codes.

Parameters and instructions are communicated between applications via using intrinsic MPI procedures. MPI_SEND Fortran subroutine writes parameter outputs or instructions from one code to the external buffer of the other, whereas MPI_RECV subroutine reads the matching parameters or instructions from the buffer. These subroutines are blocks of operations that result in the explicit coupling and synchronization of the codes by interrupting a simulation while sending or waiting for a message to the buffer. Synchronization is the temporal dependency of individual steps. MPI wraps this up in a relatively simple way as explained in Figure 4-2.

Process 1 (e.g., CTF)	Process 2 (e.g., RDFMG-NEM)
Send Msg/Data to Process 2 (Through an MPI_SEND)	Blocks code execution; Waiting for Msg/Data from Process 1
Blocks code execution; Wait for acknowledgment Msg	Receives Msg/Data
Blocks code execution; Wait for acknowledgment Msg	Sends acknowledgment Msg
Acknowledgment Msg received	Continue execution …
Continue execution …	

Figure 4-2: Synchronized Communication Representation of an MPI_SEND - MPI_RECV Pair Block.

One instruction is executed on each process's local data via a separate thread of execution. Code segments with distinct alternative variations are added into the program when it is necessary to distinguish across processes. The processes are recognized inside the execution mode when the processes must be distinguished from one another. In summary, the MPI program is a single piece of code that runs across all processes, with branching at the points where they differ from one another (Kopysov et al., 2006).

4.3 Spatial Coupling

It is critical to create appropriate nodalization and mapping schemes between the thermal–hydraulics core model, the heat-structure/rod model, and the neutronics model in both the axial and radial planes, especially since the mesh is usually different between the different codes This process is called spatial mesh overlays and is the major component in spatial coupling between different physics solvers.

In order to generate accurate solutions in a reasonable length of time, it is necessary to carefully pick appropriate spatial mesh overlays, particularly for a particular transient. This presents a difficult problem (K. Ivanov & Avramova, 2007).

There are two types of spatial coupling: fixed coupling and flexible coupling. In the case of the fixed coupling, one thermal–hydraulic channel will typically represent one neutronics assembly; however, the user will have the ability to specify the mapping schemes in the case of the flexible coupling using mapping methods.

In this thesis, flexible coupling was used since the mapping schemes are employed in the assignment of parameters between the two coupled codes. This approach allows the user to explore different spatial coupling schemes for achieving an optimal combination of accuracy/precision and efficiency as well as high-fidelity pin/sub-channel analysis. The mapping of coolant moderator properties (temperatures, density, boron concentration) are stored in a text file labelled “Table-1.map”, which also includes the CTF channel index, the axial level, the NEM node index, and the weighting for each correspondence. The mapping of fuel temperature and power are stored in another text file labelled “Table-2.map”. This second file includes the CTF heat structure index, axial level, NEM node index, and weighting. A coolant-centered channel/assembly level model set with four quarter-weighting CTF channels is shown in Figure 4-3. These channels correspond to the first active fuel node in the NEM model.

The mapping tables are produced using a developed separate mapping script in Python that reads the CTF and NEM inputs, extracts geometrical information, calculates weighting, and writes the mapping tables. The mapping script reads each line of CTF and NEM input. When a line begins with a defined string ("*"), flags are activated; these flags allow for the reading of geometrical data required for mapping. The following data are read from the CTF input:

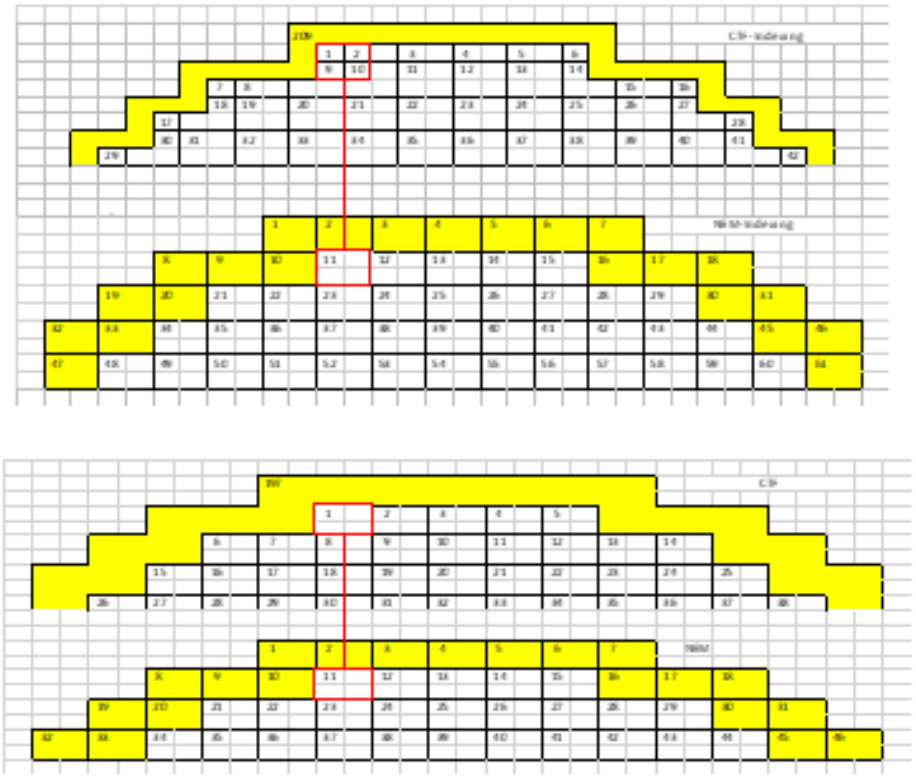
- Card 2.2 (cell location and size of the channel in the X and Y direction)
- Card 4.3 (axial cell length)
- Card 8.3 (sub-channel cell corresponding the heat structure)

- Card 8.6 (total heat structure)

The script skip lines that start with *. The script assumes that a bypass channel and bypass heat structure exist, and it is important to include these in the CTF input if they are not already there.

The data read from NEM are:

- Geometry – size (number of nodes in x, y, and z directions);
- Core layout
- Axial node sizes



- **Figure 4-3: Spatial Coupling Mapping “Table1.map” and “Table2.map” file Contents Visualized**

The axial weighting factors are defined by the axial level overlap in the two inputs. The script then constructs an array of CTF sub-channel cell edge positions, with values normalized between 0 and 1, based on the cell locations and two-dimensional size data. To compute the NEM node edges, all nodes are the same size, and non-bypass node edges are normalized between 0 and 1. In order to determine the locations of the CTF heat structure, a weighted computation is performed that takes into account the locations of the contributing sub-channel cells. The comparison of normalized CTF cell borders to normalized NEM node borders is required to determine the weighting factors that should be applied within axial levels; if there is overlap, the percentage of the total size of the NEM node is computed. the borders of each set of CTF cells are compared to the edges of each set of NEM node boundaries. To get the total cell to node weighting factor, the newly calculated factor is merged with the previously obtained axial weighting factor.

CHAPTER 5: RESULTS AND DISCUSSIONS

5.1 Code Verification and Validation

The nuclear industry has always prioritized the safe, dependable, and cost-effective operation of the nuclear power reactor fleet. Given these aims, the development, validation, and application of multi-physics predictive modeling capabilities for both normal and accident situations have progressed from best-estimate computations to first-principles high-fidelity multi-physics simulations.

The certification approach for coupled multi-physics code systems is based on the verification and validation of distinct physics models/codes, and also includes the verification and validation of the coupling methodologies of the multiple physics models (Avramova & Ivanov, 2010). The speed and capacity of computer systems continue to increase, which in turn results in users of computer models in decision making having increasingly high expectations of those models.

Verification is the process of evaluating whether a computational model created by discretizing a mathematical model of a physical event and the code used to implement the computational model can accurately represent the mathematical model of the event (Babuska & Oden, 2004) . The verification of the work that is described in this thesis will take up a greater portion of our attention.

We will verify in this Chapter the different methods implemented to accelerate the convergence rate of the coupled codes: OpenMP, CMR, and RBM. For thermal-hydraulic/neutronics coupling, the verification of developed coupled-codes system should go through a series of verifications as described in Table 2 and in reference (K. Ivanov & Avramova,

2007). The verifications include code and solution verification, and the verifications for coupled-codes systems include both for the standalone codes or for the coupled-codes system globally.

Table 2: Steps to be Followed in the Process of Verification of the Codes as well as the Coupling System

NEM Standalone	Coupling Verification	Solution Verification
• We will verify if the acceleration methods CMR and OpenMP help with NEM execution time reduction.	• The CTF/NEM model results are compared against the CTF/PARCS model results. • The coupling between the codes should be compared to check if the results are correct. • It is important also to test the communication between the codes verifying if the information that is being sending from one code to another is correct.	• Comparing the calculated values against benchmarks. The spatial homogenization equivalence may be quantified using benchmarks. • Changing some parameters, we can verify the behavior of the Rebalance method.

5.2 TMI Benchmark assembly-wise Analysis

5.2.1 Introduction

To verify and analyze the CTF/NEM coupled-codes results, it is necessary to develop thermal-hydraulic and neutronics standalone models. The thermal-hydraulic code sends to the neutronics code the following parameters: Coolant temperature and density, coolant's boron concentration, and fuel temperature. The neutronics code, with these parameters, interpolates the cross-sections from tabulated data at nominal values of the said feedback parameters and then calculates the power distribution. This power distribution of each node is then passed back to CTF or the thermal-hydraulic code. All these variables passing are allowed by the MPI-dependent modules developed in both codes as explained in Chapter 4.

The TMI-1 NPP Benchmark (K. N. Ivanov et al., 1999) is used as reference for the model. The inclusion of comprehensive three-dimensional (3-D) reactor core modeling into system transient algorithms enables "best-estimate" simulations of interactions between reactor core behavior and plant dynamics (K. N. Ivanov et al., 1999). The Nuclear Energy Agency's (NEA), Nuclear Science Committee (NSC) and the Organization for Economic Co-operation and Development (OECD) together developed a set of computer benchmark challenges for calculating reactivity transients in pressurized water reactors (PWR).

With the use of a three-dimensional neutronics core model, the PWR MSLB Benchmark is used to validate the capacity of coupled codes with coupled core-plant interactions, as well as to completely test the thermal-hydraulic coupling. This benchmark is on assembly/channel fidelity/spatial resolution level. Based on real-world plant design and operation data from the Three Mile Island – Unit 1 Nuclear Power Plant, the benchmark problem was created. The investigated transient is an MSLB in a PWR that can occur as a result of a steam line rupture upstream of the cross-connect.

5.2.2 Model Description

The reactor core is composed of 241 assemblies, being 177 fuel assemblies and 64 reflectors. The reactor core is divided axially into 24 layers with a height of: 14.88 cm, 4.71 cm, 10.17 cm, 20 nodes of 14.88 cm, 12.266 cm, 2.614 cm and 14.88cm, adding up to a total active height of 357.12 cm (K. N. Ivanov et al., 1999). The top and bottom reflector has a thickness of 21.811 cm. The core is composed of 30 different types of assemblies. The geometric data for assemblies is presented in Table 3 and the definition of the assembly types is shown in Table 4. The model includes two prompt neutron groups and six delayed neutron groups. For each composition, a complete set of diffusion coefficients and macroscopic cross-sections as a function

of the moderator density and fuel temperature was defined. These cross-sections account for scattering, absorption, and fission. Based on the reactor conditions being modeled, a linear interpolation approach is employed to generate the appropriate total cross-sections from the tabulated ones. The cross-section sets are constructed into a library of cross-sections.

Table 3: TMI Model - Fuel Assembly Geometry Data

Parameter	Value
Pellet diameter	9.391 mm
Clad diameter (outside)	10.928 mm
Clad wall thickness	0.673 mm
Fuel rod pitch	14.427 mm
Guide tube diameter (outside)	13.462 mm
Guide tube diameter (inside)	12.650 mm
Geometry	15 x 15
Number of fuel pins	208
Number of guide tubes	16
Number of in-core instrument	1

The feedback/coupling between neutronics (NEM) and thermal-hydraulics (CTF) is defined by using mapping techniques (spatial mesh overlays) in the radial and axial core planes, as described previously. We assume the reactor is at the end of its lifecycle with boron

concentration equal 5 ppm, the Xe and Sm concentrations are at equilibrium, and the average core exposure is 24,58 GWD/MT at 650 EFPD (the end of the cycle). Table 5 presents the initial conditions for TMI model at 2772MWt.

Table 4: TMI Model – Assembly Types

Assembly	Characteristics		
1	4.00w/o	No BP	No Gd pins
2	4.95 w/o	3.5% BP	4 Gd pins
3	5.0 w/o	3.5% BP pulled	4 Gd pins
4	4.95 w/o	3.5% BP	4 Gd pins
5	4.40 w/o	No BP	No Gd pins
6	5.00 w/o	3.5% BP	4 Gd pins
7	4.85 w/o	No BP	4 Gd pins
8	4.85 w/o	No BP	4 Gd pins
9	4.95 w/o	3.5% BP pulled	4 Gd pins
10	4.95 w/o	3.5% BP	4 Gd pins
11	4.85 w/o	3.5% BP pulled	4 Gd pins
12	4.95 w/o	3.5% BP	4 Gd pins
13	5.00 w/o	3.5% BP pulled	4 Gd pins
14	5.00 w/o	No BP	4 Gd pins
15	4.95 w/o	No BP	4 Gd pins
16	4.95 w/o	3.5% BP pulled	4 Gd pins
17	4.95 w/o	3.5% BP	4 Gd pins
18	4.95 w/o	3.5% BP pulled	4 Gd pins
19	5.00 w/o	3.5% BP	4 Gd pins
20	4.40 w/o	No BP	No Gd pins
21	4.85 w/o	3.5% BP pulled	4 Gd pins
22	4.40 w/o	No BP	No Gd pins
23	4.95 w/o	3.5% BP	No Gd pins
24	4.95 w/o	3.5% BP pulled	4 Gd pins
25	5.00 w/o	No BP	8 Gd pins
26	5.00 w/o	No BP	4 Gd pins
27	5.00 w/o	No BP	No Gd pins
28	4.94 w/o	3.5% BP pulled	4 Gd pins
29	5.00 w/o	No BP	4 Gd pins
30		Radial Reflector	

Table 5: Initial Conditions for TMI Model

Parameter	Value
Core Power	2772 MWt
RCS cold leg temperature	563.67 K
RCS hot leg temperature	591.43 K
RCS pressure	14.96 MPa
Core flow rate	16052.4 kg/s

The coupled model is an assembly resolved model with a total of 178 radial cells (177 fuel assemblies + 1 for the radial reflector). Axially, the core was divided into 28 meshes, 26 for the fuel active length and 2 for the top and bottom reflector. This nodalization is visualized in Figure 5-1. One representative fuel pin is modeled in CTF for each assembly with 10 radial nodes in the fuel and 2 radial nodes in the cladding.

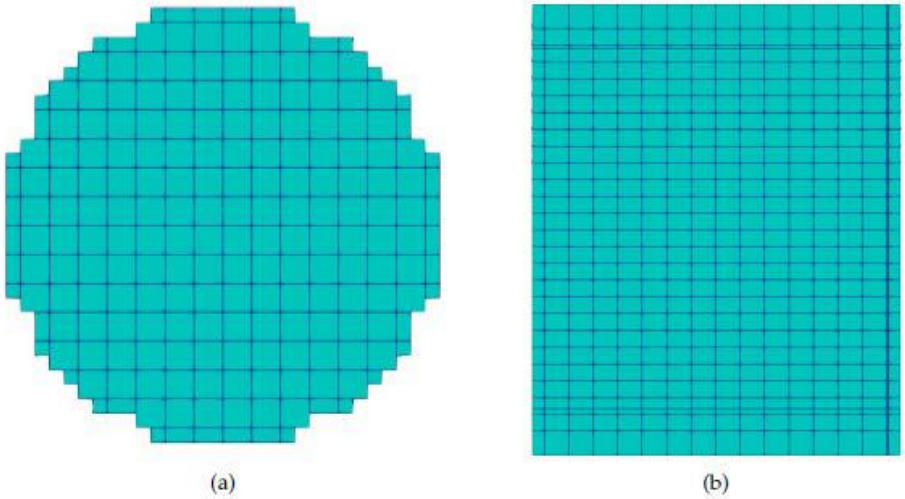


Figure 5-1: Radial (a) and Axial (b) Nodalization of the TMI CTF/NEM Model.

5.2.3 NEM Standalone Analysis

The problem setting for the steady-state simulation is based on the Pressurized Water Reactor Main Stem Line Break (MSLB) benchmark specifications (K. N. Ivanov et al., 1999), including the geometry and configuration. The MSLB model does not consider the spacer grids of the assemblies.

The coarse mesh rebalance is used in NEM code to accelerate the convergence of the outer or fission source iterations. Also, with the purpose of accelerating the convergence of the inner iterations, the OpenMP was implemented for the inner iterations in the NEM code. To compare the performance of the original code to the modified versions, we performed timing analysis simulating the TMI benchmark. The modified NEM code has both the new optimized and stabilized CMR (cf. Section 3.2.3) and the OpenMP implemented. The TMI benchmark was simulated using NEM with and without the aforementioned options, and the case was run for 100 times to get better statistics of the simulation time. The coarse mesh considered for this full-core geometry benchmark is shown in Figure 5-2, where 9 coarse nodes in both x (highlighted with thick black border) and y directions (colored with different colors) are used as the coarse mesh. The numbers inside each coarse node represent the material type for each of the fine mesh nodes.

Figure 5-3 shows the effect of applying CMR acceleration on the TMI benchmark. The CMR can reduce the simulation time by two folds, and the same behavior can be observed for the 4 OpenMP threads case without the CMR acceleration flag being on which confirms the importance of the parallelization process implemented. The average time for the two previous cases is 6.535 s and 6.407 s, respectively. For the parallel case when CMR is turned on, the acceleration is enhanced as expected, and the average simulation time is 4.237 s. The standard deviation ranged was between 0.185 s and 0.475 s for all cases. The standard deviation for the

parallel case was large since the processors are not always free considering that the computational node used is also being utilized by other RDFMG cluster users as well.

Coarse-Mesh	1	2	3	4		5							6		7	8	9
Fine-Mesh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

					30	30	30	30	30	30	30						
				30	30	30	21	15	8	15	21	30	30	30			
			30	30	28	25	20	14	7	14	20	25	28	30	30		
		30	30	29	27	24	19	13	6	13	19	24	27	29	30	30	
		30	28	27	26	23	18	12	5	12	18	23	26	27	28	30	
30	30	25	24	23	22	17	11	4	11	17	22	23	24	25	30	30	
30	21	20	19	18	17	16	10	3	10	16	17	18	19	20	21	30	
30	15	14	13	12	11	10	9	2	9	10	11	12	13	14	15	30	
30	8	7	6	5	4	3	2	1	2	3	4	5	6	7	8	30	
30	15	14	13	12	11	10	9	2	9	10	11	12	13	14	15	30	
30	21	20	19	18	17	16	10	3	10	16	17	18	19	20	21	30	
30	30	25	24	23	22	17	11	4	11	17	22	23	24	25	30	30	
		30	28	27	26	23	18	12	5	12	18	23	26	27	28	30	
		30	30	29	27	24	19	13	6	13	19	24	27	29	30	30	
			30	30	28	25	20	14	7	14	20	25	28	30	30		
				30	30	30	21	15	8	15	21	30	30	30			
					30	30	30	30	30	30	30						

Figure 5-2: Coarse Mesh used for the TMI Benchmark Problem.

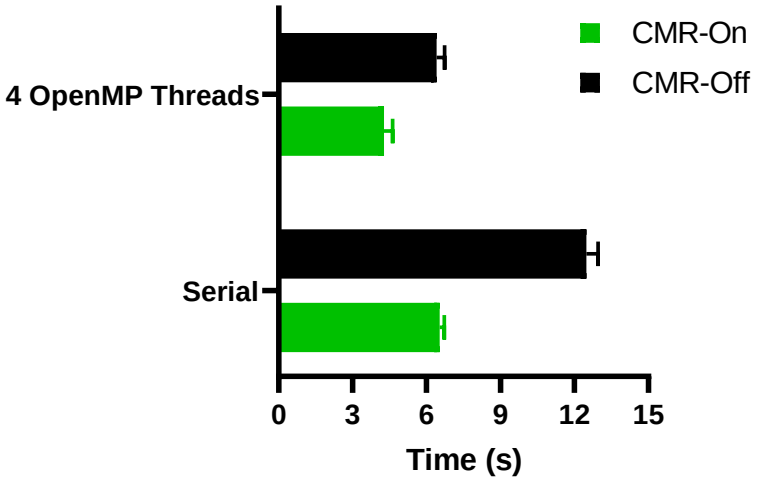


Figure 5-3: Time Comparison with Different Accelerations Measures for NEM Standalone Code Using the TMI Benchmark Problem.

The effect of using more threads for the OpenMP parallel simulation is shown in Figure 5-4. Although the 7 threads case shows the best acceleration measure, but the added time savings does not guarantee its preference over the 4 threads case due to the high standard deviation observed. The 10 threads case scaled poorly than the other two cases. This can be attributed purely to how parallel simulation works (cf. Chapter 3), and the wait time taken for each processor to synchronize with one another. If one of the processors is occupied by another task (i.e., on the same computational node on the cluster), then they will hinder the processors group progress as is shown in the high standard deviation observed for the 10 threads case.

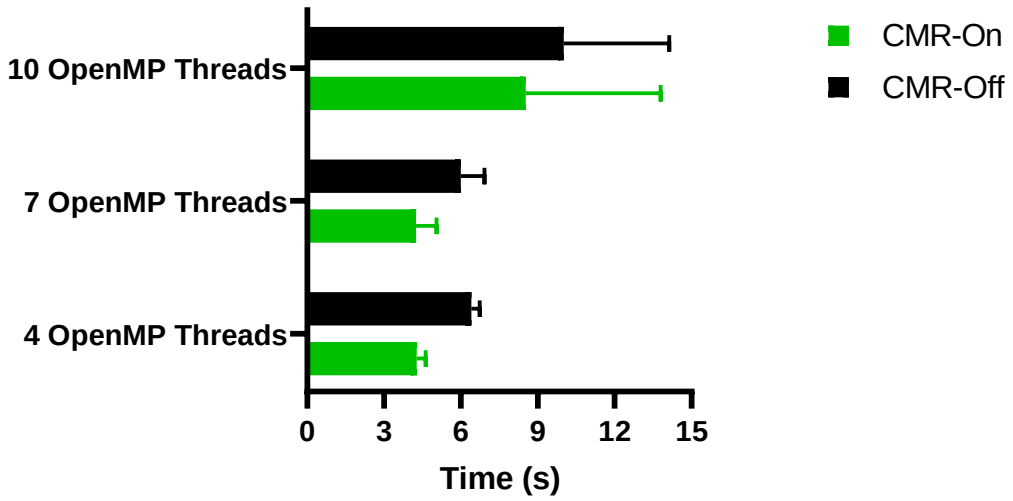


Figure 5-4: Time Comparison Between Different Values for OpenMP Threads Using the TMI Benchmark Problem.

As mentioned in Section 3.2.3, the CMR method was stabilized and enhanced. To see the effect of this enhancement, Figure 5-4 is replotted in Figure 5-5 considering both the old CMR procedure and the new enhanced one inside NEM. The average simulation time was reduced by around 17% while the difference in the k-eff computed by both of the CMR cases was beyond pcm

significance (exactly 0.007 pcm). The same behavior in Figure 5-3 following increasing the number of threads is shown in Figure 5-4 as well, leading to the necessity of using a dedicated computational node for the simulation.

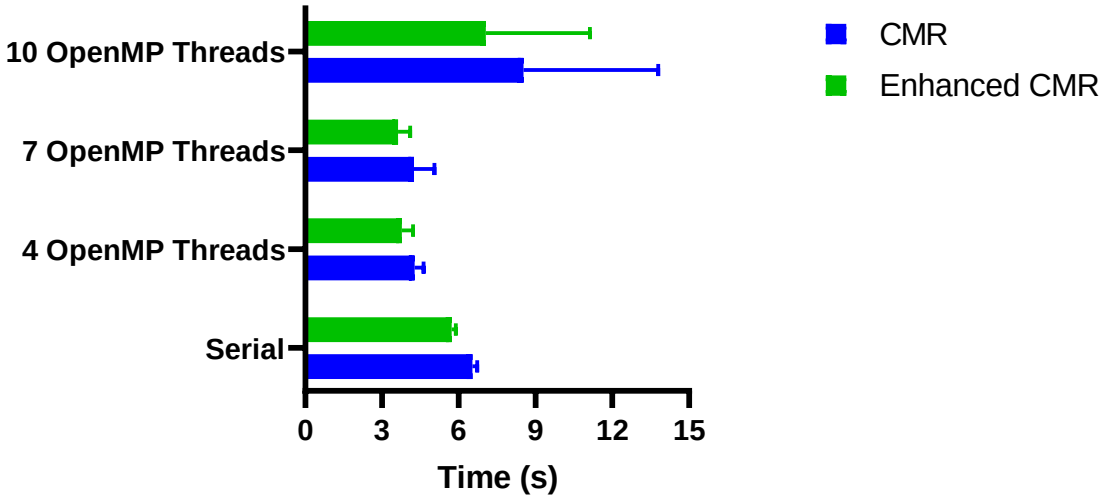


Figure 5-5: Time Comparison with Enhanced CMR Solver for NEM Standalone Code Using the TMI Benchmark Problem.

5.2.4 Coupled Analysis

The CTF/NEM results were compared against the CTF/PARCS to verify if the results of the model were correct. This is an important test to verify if the information between the codes is being sent and received correctly.

5.2.4.1 Performance of the CMR and OpenMP for Assembly Wise Geometries

In this section, we analyze the performance of the acceleration methods that can be used to accelerate the convergence rate, especially on the neutronic code (i.e., NEM) side. For acceleration of NEM, the CMR methodology initially present in NEM was improved and also the parallelization methodology OpenMP was implemented, as described in detail in Chapter 3. At

first, the coupling methodology was carried out such that a previous neutronics-iteration solution was not used as an initial value for the next coupled iteration (that is, the neutronics solution was initialized when a thermal hydraulics solution is received from CTF). Figure 5-6 shows the effect of applying the different acceleration measures and the reduction in the simulation time observed. The OpenMP and CMR when applied together shows the best acceleration and thus reduction in the simulation time.

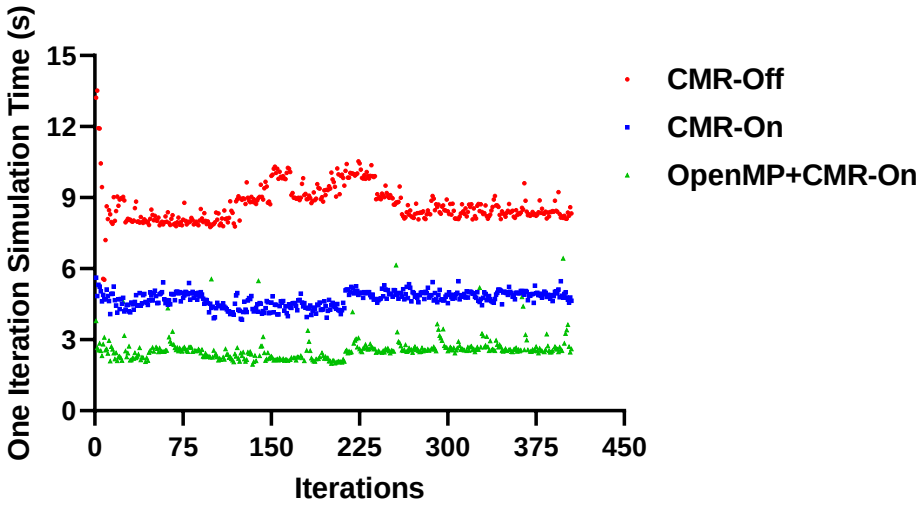


Figure 5-6: Applying NEM Acceleration Measures to the Simulation of the TMI-Benchmark Coupled Case.

The total coupled-simulation time taken without any acceleration measure is reported in Table 6. This coupled-simulation total time was 81 minutes (of which NEM took 58 minutes). Applying CMR reduced the time to 60 minutes (of which NEM took 41 minutes) and adding OpenMP to NEM made the total coupled-simulation time be 41 minutes (of which NEM took 22 minutes).

Table 6: Total Coupled-Simulation Time for NEM and CTF with OpenMP and CMR.

Acceleration Measure	Total Simulation Time	NEM Simulation Time
No Acceleration	81 minutes	58 minutes
CMR Only	60 minutes	41 minutes
OpenMP + CMR	41 minutes	22 minutes

Another possibility is to not reinitialize the NEM converged fission source and to use the one from the previous iteration directly to start each coupled iteration. This is the case in the semi-log Figure 5-7, which shows (like Figure 5-6) the coupled-simulation time for each iteration and for each acceleration measure used. The total coupled-simulation iteration time for this case was around 29 minutes, which then proves that not reinitializing the solver does reduce the total simulation time. Table 6 shows the total steady state solver simulation time as well as the total simulation time for NEM for each acceleration measure. The difference between the two equals the cross-sections setting time (e.g., cross-section interpolation and printing to an external file) which averages 0.5 seconds per iteration.

Table 7 and Figure 5-7 reveal that the cases with no CMR show better acceleration behavior and less simulation time for NEM. We can observe that for the very first 10-20 iterations, the CMR and OpenMP has managed to reduce the simulation time from 12 seconds (with no acceleration measures) to approximately 4 seconds before the fission source enters the convergence regime that lasts until the end of the simulation. In this region, we have a converged source from the previous iteration and NEM can solve with very few outer iterations that CMR will not help in this regard. The CMR solution actually takes more time to converge in this case than the nearly constant cross-section setting time (around 0.5 seconds) and this translates to higher total simulation time eventually.

Table 7: Total and Solver Simulation Times in Seconds for NEM with Different Acceleration Measures.

Acceleration Measure	Total Simulation Time	Solver Simulation Time
No Acceleration	588.60	340.46
CMR Only	641.85	390.50
OpenMP + CMR	550.20	290.20
OpenMP Only	**441.52**	**177.01**

Mathematically, the CMR method reduces the numerical error (or modes) in the solver matrix (i.e., not only geometrically). When this error is surpassed in the convergence regime, there is no point of it to be used there (e.g., in the coupled cases like these). When the CMR is turned on, it will introduce a different error than the one converged, resulting in more time to reduce this error. Moreover, the CMR method turns on after 6 outer iterations (a requirement of the method to have a good initial guess for the out-currents), which also attributes to the time taken. OpenMP method (parallelization) has a better performance during the entire process of calculation. The implementation of the OpenMP method in NEM code is a substantial improvement on the NEM code as an acceleration mechanism.

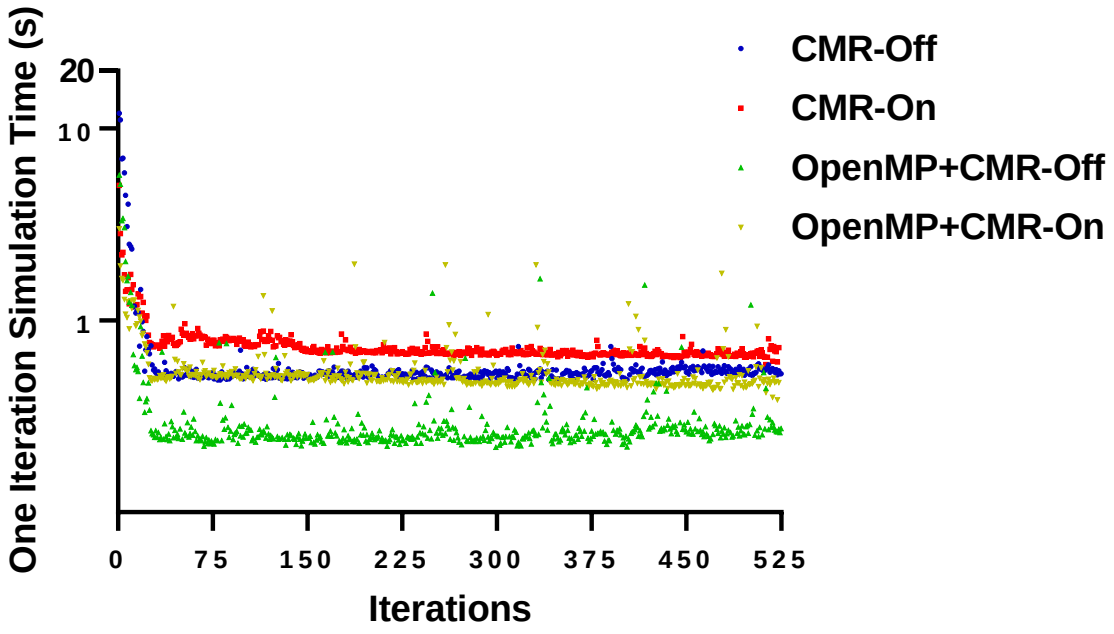


Figure 5-7: Performance of the Different Acceleration Measures for the Coupled-Simulation of the TMI-Benchmark Problem.

5.2.4.2 Comparison between both NEM/CTF and PARCS/CTF Results

To verify the CTF/NEM solution results, CTF/PARCS was used as described in Section 5.1. The k-eff difference between both codes was less than 18 pcm. Figure 5-8 represents the radial relative power distribution determined by the CTF/NEM code, and Figure 5-9 shows the radial relative power determined by CTF/PARCS code. The maximum absolute relative difference found is 0.64%, and it is depicted in Figure 5-10. Figure 5-11 presents the axial power as determined by CTF/NEM code and CTF/PARCS code.

Figure 5-8: 2D Radial Power Profile Computed by CTF/NEM.

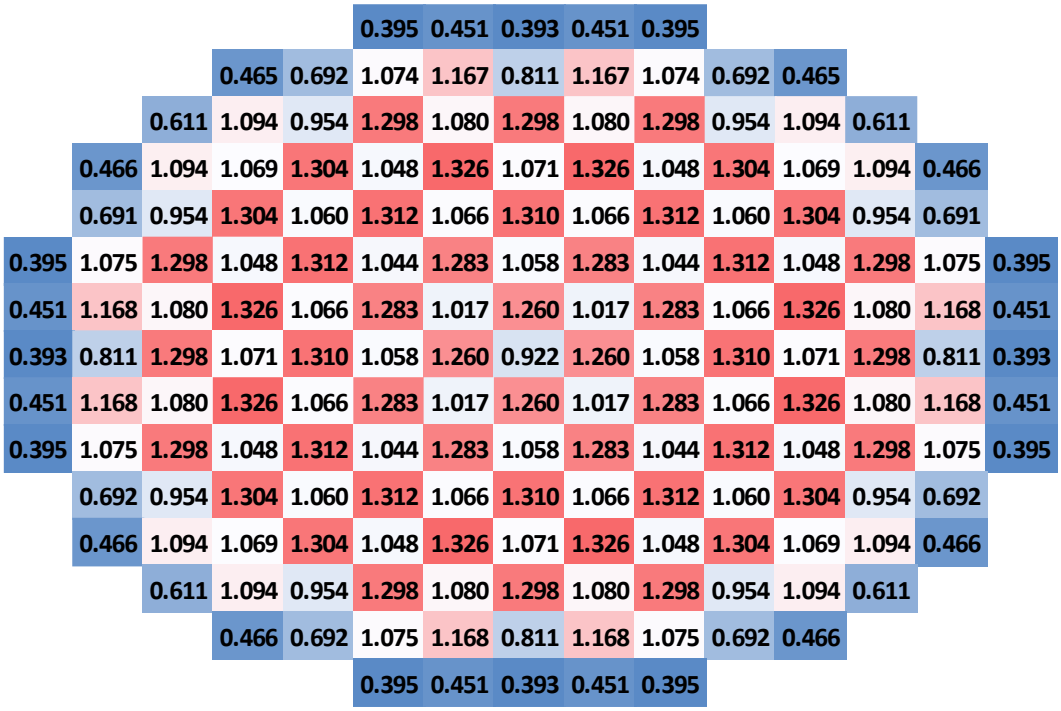


Figure 5-9: 2D Radial Power Profile Computed by CTF/PARCS.

				-0.27%	-0.19%	-0.08%	-0.19%	-0.27%					
			-0.46%	-0.37%	-0.04%	-0.01%	0.00%	-0.01%	-0.04%	-0.37%	-0.46%		
		-0.64%	-0.28%	-0.07%	-0.01%	0.12%	0.10%	0.12%	-0.01%	-0.07%	-0.28%	-0.64%	
	-0.51%	-0.27%	-0.05%	0.04%	0.08%	0.18%	0.31%	0.18%	0.08%	0.04%	-0.05%	-0.27%	-0.51%
	-0.36%	-0.07%	0.04%	0.25%	0.21%	0.24%	0.24%	0.24%	0.21%	0.25%	0.04%	-0.07%	-0.36%
-0.30%	-0.05%	-0.01%	0.08%	0.21%	0.26%	0.30%	0.24%	0.30%	0.26%	0.21%	0.08%	-0.01%	-0.05%	-0.30%
-0.20%	-0.02%	0.12%	0.18%	0.24%	0.29%	0.27%	0.28%	0.27%	0.29%	0.24%	0.18%	0.12%	-0.02%	-0.20%
-0.08%	-0.01%	0.09%	0.30%	0.24%	0.24%	0.28%	0.39%	0.28%	0.24%	0.24%	0.30%	0.09%	-0.01%	-0.08%
-0.20%	-0.02%	0.11%	0.18%	0.24%	0.29%	0.27%	0.28%	0.27%	0.29%	0.24%	0.18%	0.11%	-0.02%	-0.20%
-0.30%	-0.05%	-0.02%	0.08%	0.21%	0.26%	0.29%	0.24%	0.29%	0.26%	0.21%	0.08%	-0.02%	-0.05%	-0.30%
	-0.37%	-0.08%	0.03%	0.25%	0.21%	0.24%	0.24%	0.24%	0.21%	0.25%	0.03%	-0.08%	-0.37%	
	-0.51%	-0.28%	-0.05%	0.03%	0.08%	0.18%	0.30%	0.18%	0.08%	0.03%	-0.05%	-0.28%	-0.51%	
		-0.64%	-0.28%	-0.08%	-0.02%	0.11%	0.09%	0.11%	-0.02%	-0.08%	-0.28%	-0.64%	
			-0.51%	-0.37%	-0.05%	-0.02%	-0.01%	-0.02%	-0.05%	-0.37%	-0.51%		
				-0.30%	-0.20%	-0.08%	-0.20%	-0.30%					

Figure 5-10: Relative Difference for the 2D Radial Power Profile Between both Codes.

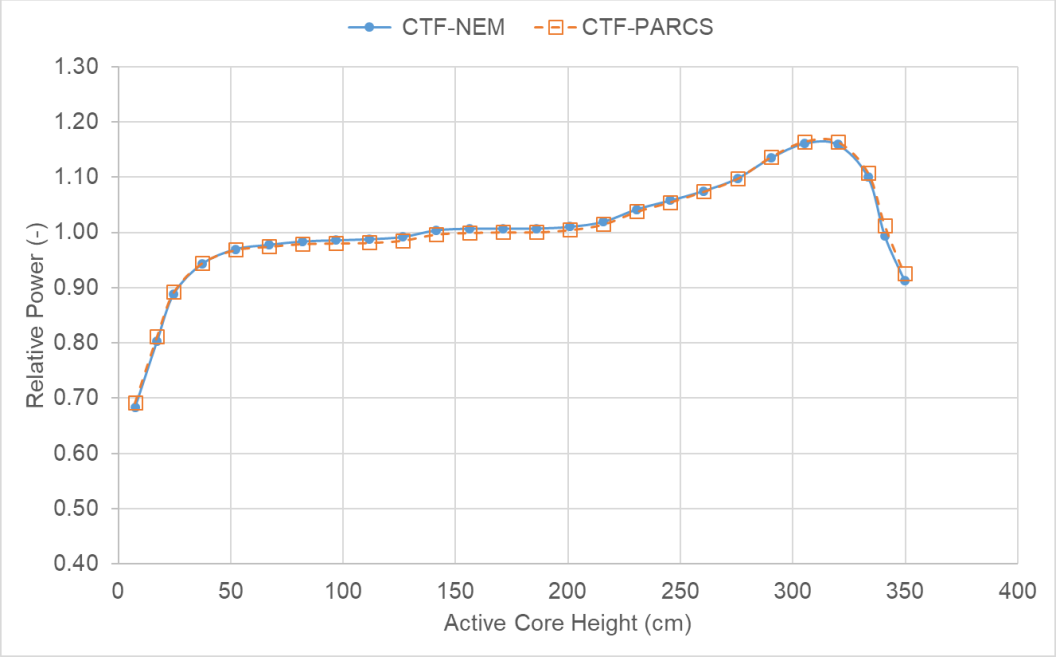


Figure 5-11: 1D axial Power Profile Comparison Between both Codes.

Figure 5-12 presents the radial fuel temperature determined by the CTF/NEM code, and Figure 5-13 the radial fuel temperature determined by CTF/PARCS code. The maximum relative difference found is 0.41%, and it is depicted in Figure 5-14. Figure 5-15 presents the axial temperature determined by NEM code and PARCS code.

Finally, Figure 5-16 presents the radial core outlet coolant density (kg/cm3) determined by CTF/NEM coupled code, and Figure 5-17 the radial core outlet coolant density determined by CTF/PARCS. The maximum relative difference found is 0.04%, and it is depicted in Figure 5-18. Figure 5-19 presents a comparison of the axial coolant density distribution predicted by both coupled codes.

				728.80	749.07	728.61	749.07	728.80						
			754.45	830.64	941.88	966.34	869.20	966.34	941.88	830.64	754.45			
		804.11	946.69	911.51	1000.35	947.43	1000.74	947.43	1000.35	911.51	946.69	804.11		
	754.60	946.69	943.47	1002.27	939.32	1008.05	945.80	1008.05	939.32	1002.27	943.47	946.69	754.60	
	830.61	911.53	1002.27	942.78	1004.66	944.26	1004.23	944.26	1004.66	942.78	1002.27	911.53	830.61	
728.90	941.91	1000.36	939.33	1004.66	938.57	997.88	941.94	997.88	938.57	1004.66	939.33	1000.36	941.91	728.90
749.10	966.37	947.45	1008.06	944.26	997.88	930.72	992.00	930.72	997.88	944.26	1008.06	947.45	966.37	749.10
728.62	869.22	1000.75	945.81	1004.23	941.95	992.00	904.29	992.00	941.95	1004.23	945.81	1000.76	869.22	728.62
749.11	966.37	947.45	1008.07	944.26	997.88	930.72	992.00	930.72	997.88	944.26	1008.07	947.45	966.38	749.11
728.89	941.91	1000.37	939.34	1004.67	938.57	997.89	941.95	997.89	938.57	1004.67	939.34	1000.37	941.91	728.89
	830.64	911.54	1002.28	942.78	1004.67	944.26	1004.23	944.26	1004.67	942.78	1002.28	911.54	830.64	
	754.59	946.71	943.48	1002.28	939.34	1008.07	945.82	1008.07	939.34	1002.28	943.48	946.71	754.59	
		804.13	946.71	911.54	1000.37	947.45	1000.76	947.45	1000.37	911.54	946.71	804.13		
			754.59	830.64	941.92	966.38	869.22	966.38	941.92	830.64	754.59			
				728.90	749.11	728.63	749.11	728.90						

Figure 5-12: 2D Radial Fuel Effective Temperature Computed by CTF/NEM.

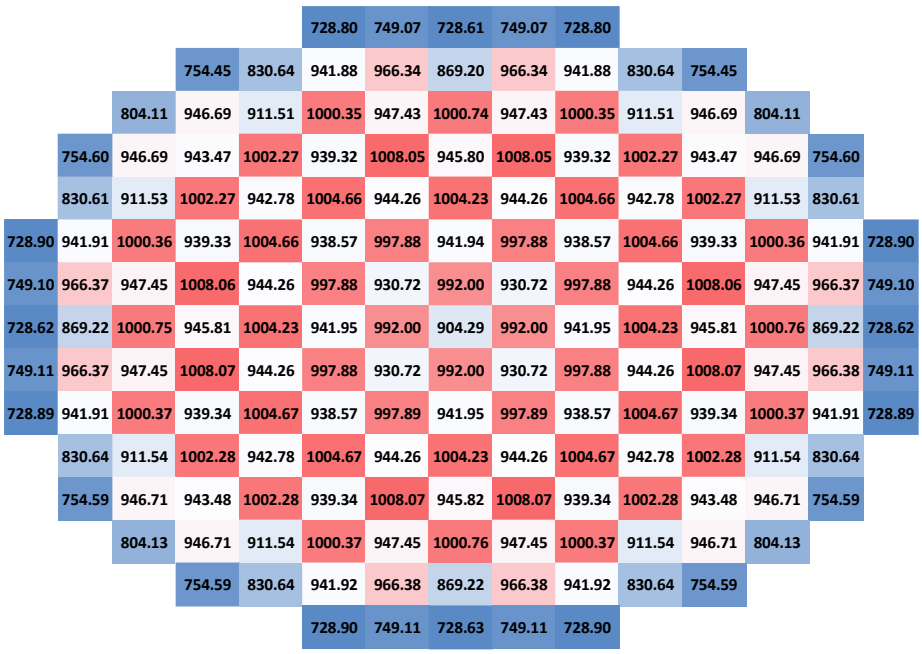


Figure 5-13: 2D Radial Fuel Effective Temperature Computed by CTF/PARCS.

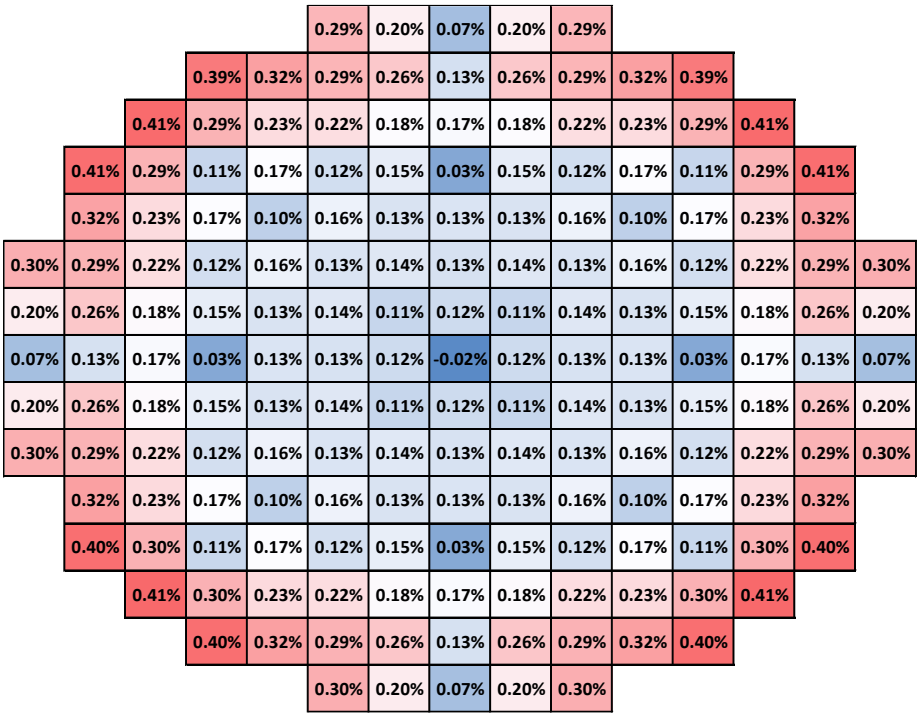


Figure 5-14: Absolute Relative Difference for the Radial Fuel Effective Temperature Between Both Codes.

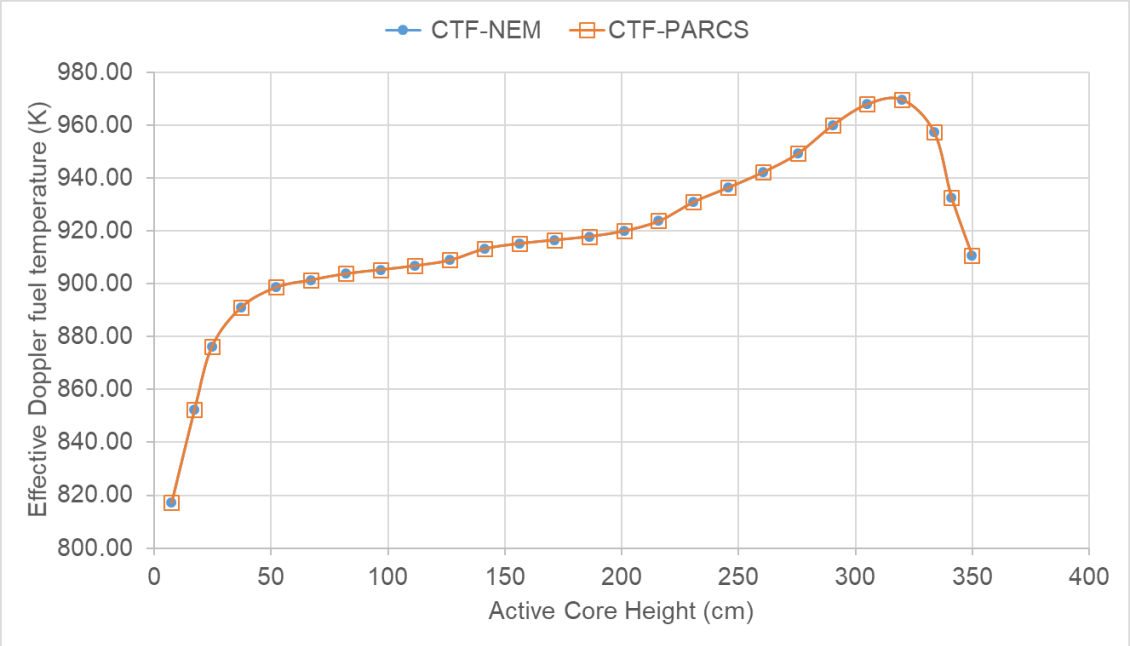


Figure 5-15: Axial Fuel Effective Temperature Computed by CTF/NEM and CTF/PARCS.

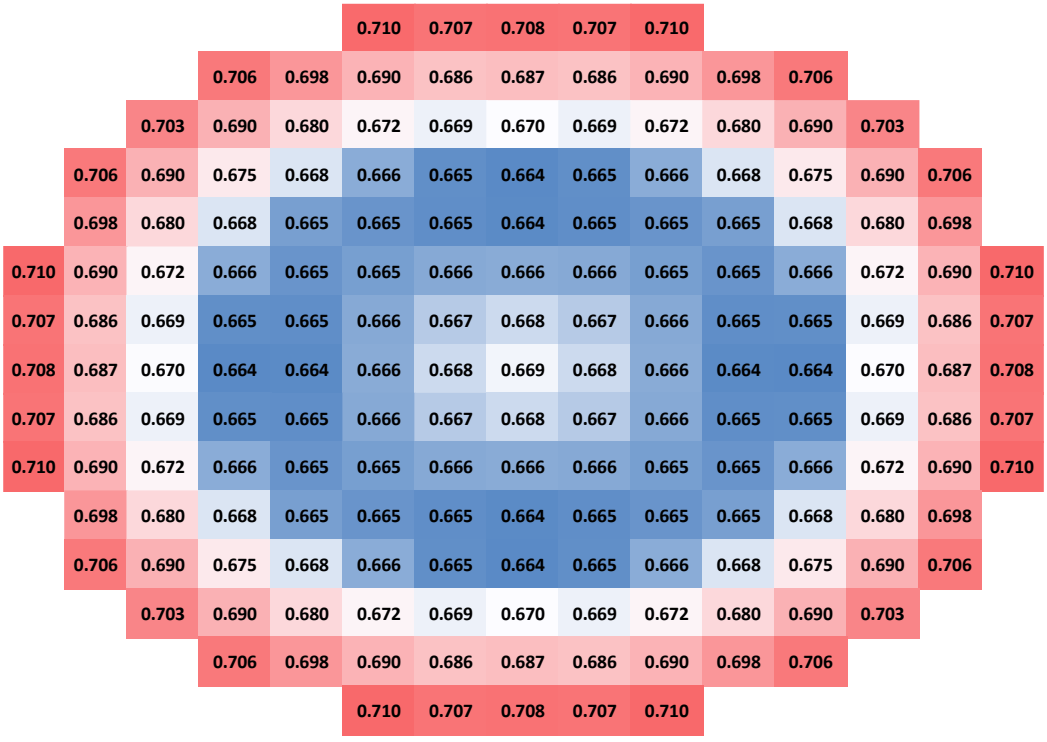


Figure 5-16: 2D Radial Coolant Density (g/cm3) Computed by CTF/NEM.

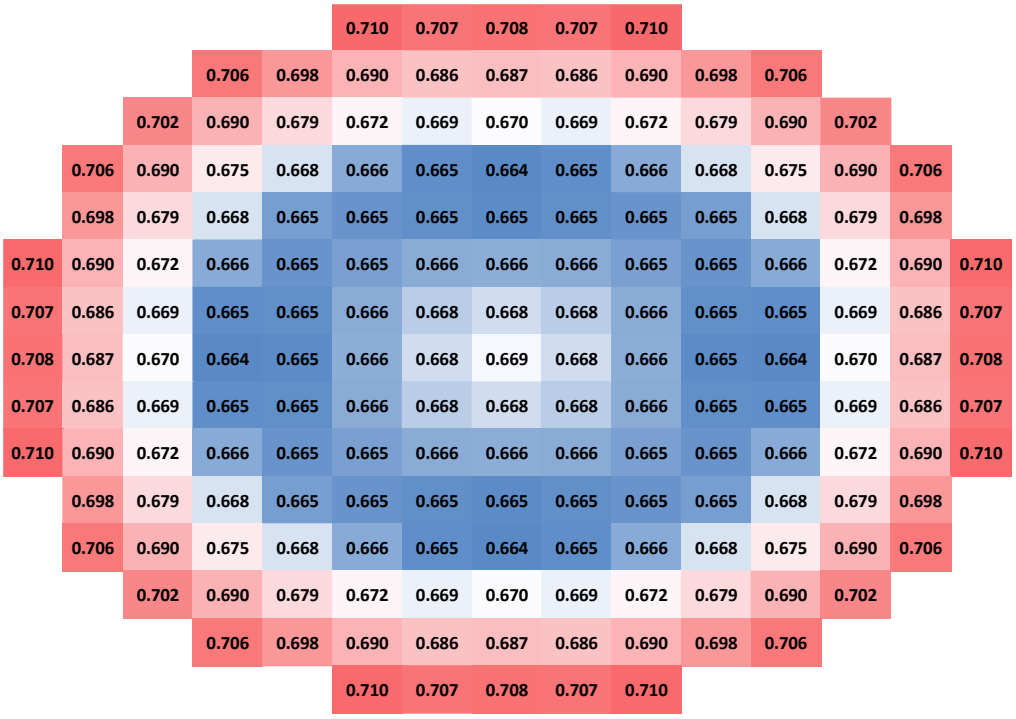


Figure 5-17: 2D Radial Coolant Density (g/cm3) Computed by CTF/PARCS.

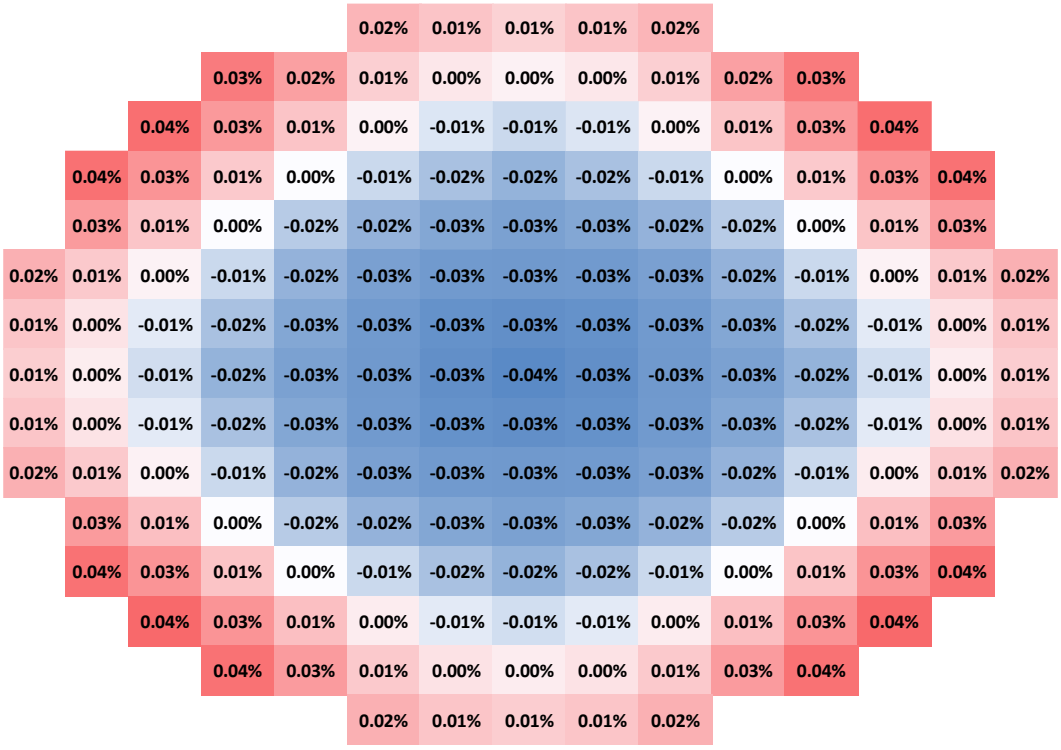


Figure 5-18: Relative Difference for the Radial Coolant Density Between Both Codes.

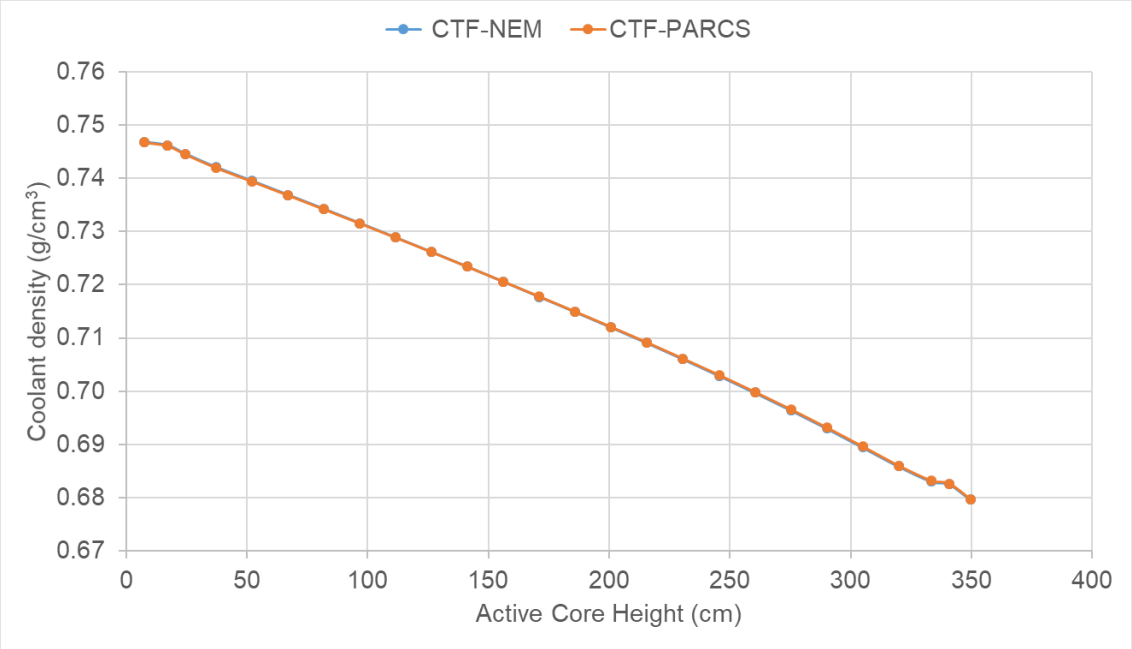


Figure 5-19: Axial Coolant Density (g/cm3) Computed by CTF/NEM and CTF/PARCS.

5.2.4.3 Performance of the Solution Interruption and Residual Balance Methods

The section presents a comparative analysis of the convergence when different methods presented in Chapter 3 to speed up the coupled code steady-state convergence. Two methods, solution interruption and residual balance, have been applied over the TMI steady-state coupled case scenario. The results presented show the evolution of CTF's residual (solver S1), and NEM's residuals (solver S2) over the simulation. The residual balance shows its capability to reduce the residual of both solvers altogether towards the convergence. Picard iteration methods has not been added to the comparison because of its the small convergence rate. This poor convergence rate drives prohibitive simulation times for pure Picard iteration method with no relaxation in the convergence criteria.

Figure 5-20 presents the evolution of the residuals when the solution interruption is applied to NEM (S2). A maximum of 5 outer iterations are performed in NEM before stopping and sending

the feedback to CTF. The convergence criteria of CTF is dynamically changed from 5.0E-3 to 1.0E-3 to achieve a faster convergence, so the S1 convergence criteria is relaxed into two steps.

Figure 5-21 presents the evolution of the residuals applying solution interruption method in both solvers, CTF (S1), and NEM (S2). A maximum of 100 iterations are performed in S1 before sending the thermal-hydraulic feedback to NEM, and in the same way 5 outer iterations are performed in NEM before stopping and sending the power feedback back to CTF. The simple solution interruption method presented before outperforms this double solution interruption method, see Table 7, because fewer number of global iterations are need.

Figure 5-22 presents the evolution of the residuals when residual balance is applied. As can be observed in Table 8, the residual balance outperforms the solution interruption method despite a higher number of global iterations needed to reach the steady-state convergence. This is due to the fact of each constitutive iteration of S1 and S2 is faster because only a few numbers of outer iterations are needed to reach the proposed termination criteria. Must be noted that each individual physics termination criteria depends on the residual of each other of the coupled physics, so the convergence is progressive in both solvers.

Table 8: Number Coupled Global Iterations and Run Time for the Coupled Convergence.

Acceleration Method	Number of global iterations	CTF/NEM Simulation Time
Solution Interruption S2	18	12 minutes 55 seconds
Solution Interruption S1 and S2	26	16 minutes 26 seconds
Residual Balance	48	9 minutes 44 seconds

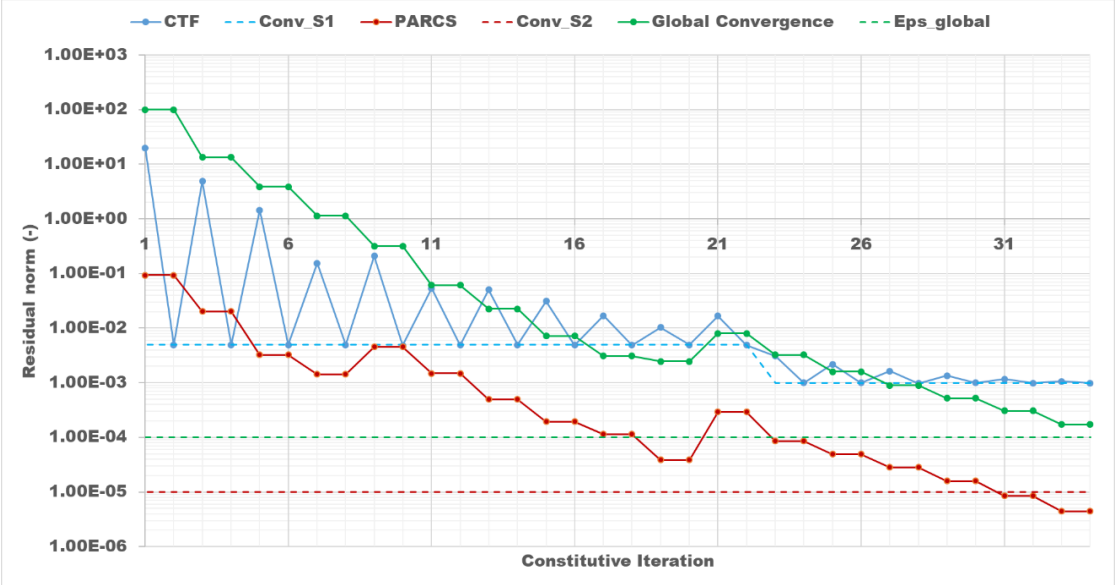


Figure 5-20: Evolution of the Residual During the Convergence with the Solution Interruption Method in S2.

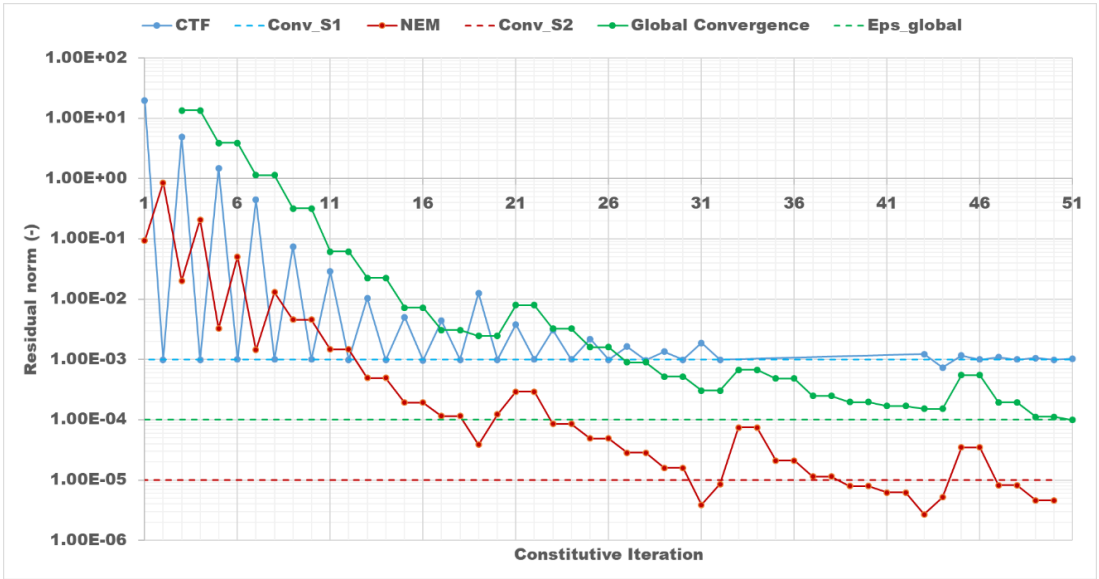


Figure 5-21: Evolution of the Residual during the Convergence with the Solution Interruption of S1 and S2 Method.

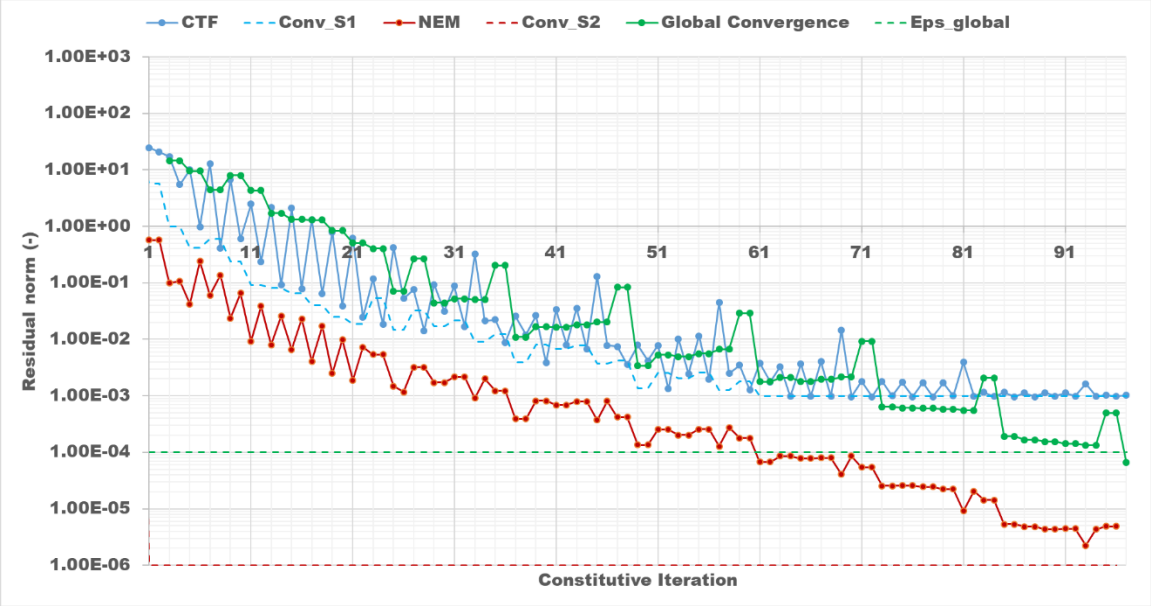


Figure 5-22: Evolution of the Residual during the Convergence with the Residual Balance Method.

5.3 C5G7-TD Analysis

The degree of precision that may be achieved by homogenization procedures is one of the most significant challenges posed by deterministic transport methods for whole-core reactor simulations (Hou et al., n.d.). Because of the restricted capabilities of computers at the time, it was not possible to perform a direct computation for whole-core heterogeneous geometries. When trying to compress the spatial heterogeneities into a tractable homogenous description, one had no choice but to rely on homogenization procedures. However, these homogenization approaches can add substantial error into the flow distribution and subsequently response rates in the homogenized zone can be significantly in error. With today's computing power, direct calculations involving the entire heterogeneous core are becoming an increasingly viable option.

In this perspective, an OECD/NEA benchmark problem was developed to test the capability of contemporary deterministic transport methods and codes to deal with problems

similar to those found in reactor cores without resorting to spatial homogenization (Hou et al., n.d.). That Benchmark was used to implement the C5G7-TD model described in this thesis.

5.3.1 Model Description

A pin resolved model representing four 17×17 fuel assemblies and 5 reflector positions is developed. Both UO2 and MOX fuel assemblies have 264 fuel pins, 24 guide tubes for control rods, and one instrument tube for a fission chamber in the middle grid-cell. This model leads to a total of 1156 pin cells plus 5 big channels for the mini-core bypass in CTF. Axially, the core was divided into 28 meshes, 26 for the fuel active length and 2 for the top and bottom reflector. This nodalization is visualized in Figure 5-23. As on TMI model, the fuel rods are modeled using 10 radial nodes in the fuel and 2 radial nodes in the cladding for the fuel conduction. The position of the spacer grids (Rubin et al., 2010) Figure 5-24. The quarter core radial symmetry is presented in the 2-D configuration (Figure 5-25). The active height of the fuel assembly is 3657.6 mm, and it is discretized into 26 axial nodes. The reflector surrounding the active core the equivalent width of one FA (214.2 mm) on the sides and top and bottom. Vacuum boundary condition has been applied to the axial boundary of the core. All the boundary condition for the core is set to zero incoming flux. Table 9 summarizes fuel assembly and core data for modeling. summarizes fuel assembly and core data for modeling.

In C5G7-TD model, each 17x17 assembly contains six intermediate spacer grids and two end grids (Godfrey, 2014) which reduce rod vibration and bow, as well as offer lateral structure support and, in some situations, coolant flow mixing. To restrict neutron absorption, the intermediate grids have been constructed out of Zircaloy-4 and are positioned within the active fuel region. The ends grids are built of Inconnel and are positioned at the very end of the fuel stack or outside of it for the purpose of providing enhanced structural support. The axial movement of

the grids is restricted by the presence of a set of spacer sleeves in each grid. These spacer sleeves make contact with the instrument tube and the guide tubes. Table 10 presents the spacer grids position, as well as the loss coefficient and the height of the grids.

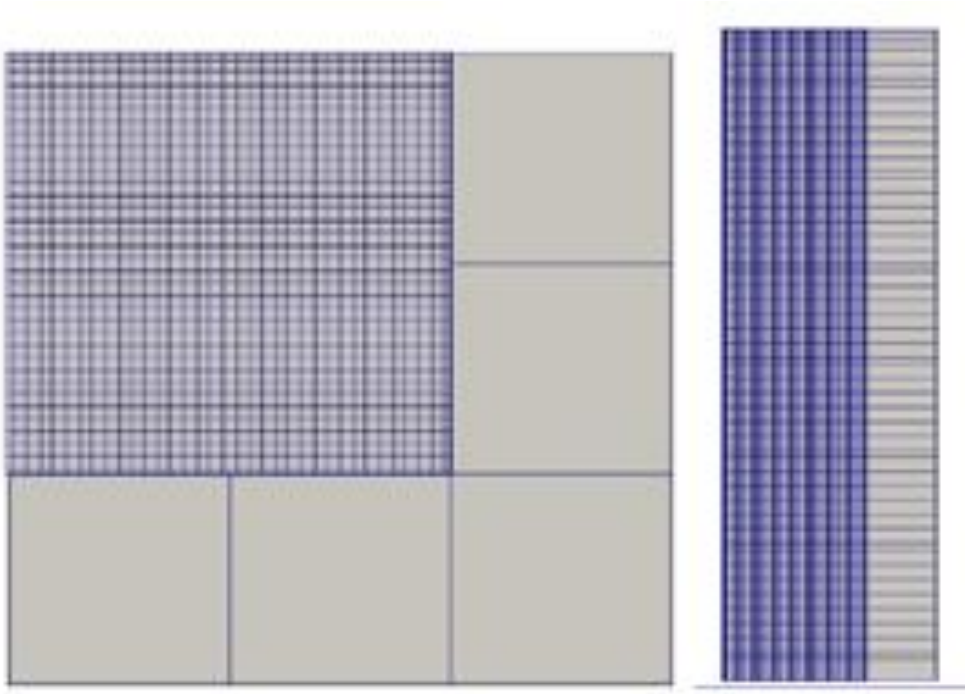


Figure 5-23: Radial and Axial Nodalization of the C5G7 CTF/NEM Model.

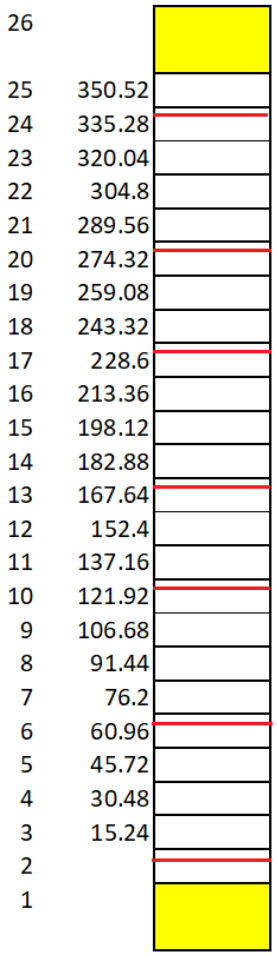


Figure 5-24: Spacer Grids Axial Positions and Axial Nodes Equivalence to Elevation.

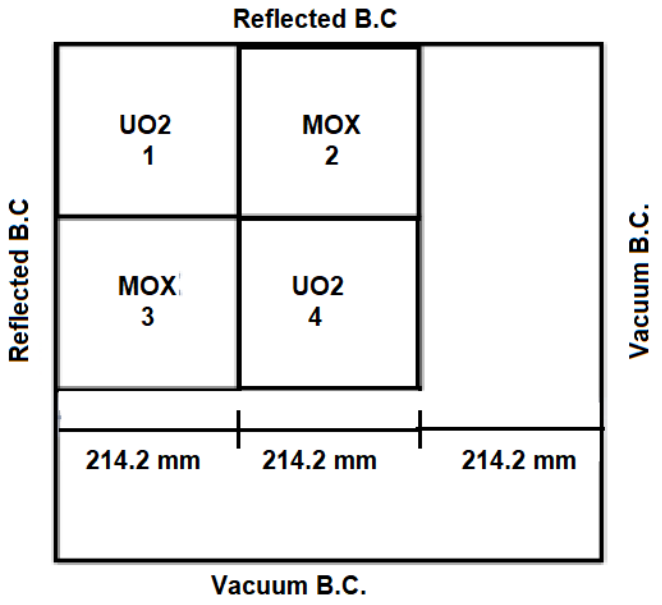


Figure 5-25: 2D Configuration for C5G7-TD Problem.

Table 9: Fuel Assembly/Core Data.

Parameter	UO2/MOX
System pressure (MPa)	15.5
Active core flow (kg/s)	328.485
Bypass flow (kg/s)	23.3224
Axial elevation of top/bottom reflector (mm)	214.2
Power (MWth)	73.886
Inlet Temperature (K)	565.15
Active Height (mm)	3657.6
Rod Pitch (mm)	12.60
Assembly pitch (mm)	214.2
Fuel pellet diameter (mm)	7.902
Fuel rod clad OD (mm)	9.166
Fuel Rod clad ID (mm)	8.02
Instrument tube material	Zircaloy
Instrument tube OD (mm)	12.064
Instrument tube ID (mm)	11.248

Table 10: Spacer Grid Positions, Loss Coefficients and Height of the Grids.

	End Grids	Intermediate Grids
Number	2	6
Material	Inconel-718	Zircaloy-4
Mass (g)	1017	875
Mixing Vanes	No	Yes
Height (mm)	38.66	38.10
Axial Locations (mm) (center of the inner strap relative to top of lower core plate)	138.84	752, 1274.0, 1796.0, 2318.0, 2840.0, 3362.0
Loss Coefficient	0.7	1.0

5.3.2 Cross-Sections

Cross-sectional measurements are used to quantify how much neutrons interact with nuclei. The nuclear cross-section of a nucleus is used to describe the probability that a nuclear reaction will occur. Scattering reactions and absorption reactions are two types of interactions that can take place between neutrons and other particles inside of a nuclear reactor. Elastic scattering makes use of some of the kinetic energy in activating a target nucleus, while inelastic scattering does not. Scattering can be further divided into two categories: elastic scattering and inelastic scattering. During the absorption process, the key reactions that take place are fission, capture, and neutron emission. In the computations that are carried out in relation to lattice physics, a wide variety of cross-sections, such as total cross-section, capture, absorption, scattering, and gamma cross-sections, are applied.

Kan Ni (Ni & Hou, n.d.) developed an improved iterative technique that is based on the Hi2Lo scheme. This technique has the capacity to produce transport-like reactor physics calculation results at a much-decreased computation cost. An effective acceleration mechanism was given as a means of accomplishing this goal. According to this mechanism, high-fidelity lattice physics calculations in the early phases of the Hi2Lo scheme would be replaced with

calculations of intermediate fidelity. This made it possible to produce converged pin-wise flux distribution, which has been demonstrated to be essential to the generation of correct homogenized cross-sections, in a way that is significantly more cost effective and, as a result, reduces the overall computing cost. The C5G7-TD model that is employed in this thesis takes advantage of the cross-sections (Ni & Hou, n.d.) that were obtained by that technique and presented a good result.

5.3.2.1 Description of the Cross Section Generation and Modeling Model

Fuel assembly models with infinite lattice geometry (using reflective boundary conditions) were prepared for the SCALE 6.2 TRITON/NEWT lattice physics calculation sequence to generate the cross-sections. In the generation process, the fully heterogeneous geometry is modelled while the outer aluminum cladding of fuel rod is removed. In this case, the material inside of each fuel pin will be fuel, zirconium cladding and moderator. The void located between the fuel and cladding is removed by expanding the fuel while preserving the total particle number. This assembly calculation model with reflective boundary condition was used by TRITON/NEWT for lattice calculations to develop pin-homogenized cross-section parametrization/modeling.

For each reactor state part of the cross-section modeling, three lattice calculations were conducted, which are for MOX assembly, UOX assembly and UOX assembly with control rod inserted. After the three lattice calculations are computed, the output files are combined and converted into nemtab format. Three thermal-hydraulic feedback variables were involved, and they are the fuel temperature, the moderator temperature and the moderator density. The reactor states are as found in Table 11

Table 11: The Reactor States Involved in Generating the Cross-Section Library.

State Variable and Unit	State Values
Fuel Temperature (K)	500, 600, 700, 800, 900, 1000, 1100, 1200, 1300
Moderator Temperature (K)	300, 400, 500, 600, 700
Moderator Density (Kg/m^3)	600, 650, 700, 750, 800

The number of generated cross-section sets is 225. The multi-group cross-sections are collapsed to 7-group energy structure format used in the above-described pin-homogenized cross-section library, and the detailed structure can be found in Table 12.

Table 12: The Energy Group Structure used for Collapsing the Cross-Sections.

Group Number	Upper Energy (eV)
1	2.0000E+07
2	1.3560E+06
3	9.5000E+03
4	5.3400E+01
5	4.1000E+00
6	6.2500E-01
7	1.2500E-01
	1.0000E-05

5.3.2.2 Cross-Section Library

The data structure of the nemtab format of the generated pin-homogenized cross-section library is shown in Table 13, and it is generated for 8 cross-section sets.

Table 13: The Cross-Sections Sets found inside the nemtab File.

Set Number	Description
1	UOX pin cells (applied to all UOX pin cells)
2	MOX 8.7% pin cells (applied to all MOX 8.7% pin cells)
3	MOX 7% pin cells (applied to all MOX 7% pin cells)
4	MOX 4.3% pin cells (applied to all MOX 4.3% pin cells)
5	Guide-tube pin cells (applied to all guide-tubes pin cells)
6	Fission chamber pin cells (applied to fission chamber pin cells)
7	Water pin cells (applied to all reflector pin cells)
8	Control rod pin cells (applied to all control rod pin cells)

The nemtab files (see Appendix-A) starts with a brief description of feedback parameters in line 4, which states the total number of feedback parameters and their states. The nemtab was generated by using the following loop structure. The 1st loop is for the spatial node (spatial-loop), which is the cross-section sets as described in the index list in Table 11. In each of the cross-section sets, the group-wise data is presented as a list from group 1 to group 7, which is the 2nd loop. Inside of each 2nd loop (group-wise loop), several cross-section data tables are provided according to the following list:

- Diffusion coefficient table for the current group,
- Absorption cross-section for the current group,
- Fission cross-section for the current group,
- Fission cross-section times the average number of prompt neutrons produced per fission for the current group,
- Scattering cross-section from the current group to the other groups,

- Discontinuity factor for the west boundary for the cell. (All filled with 1.0 in the provided cross-sections),
- Discontinuity factor for the south boundary for the cell. (All filled with 1.0 in the provided cross-section),
- Detector data for the current group, table is filled with zeros, and
- Detector data for the current group's fission events, filled with zeros.

Inside of each cross-section data table, we have the feedback points. The first 9 data correspond to the fuel temperature points, the second 5 data correspond to the moderator temperature and the last 5 data correspond to the moderator density. After the 2nd loop (group-wise) data is provided, a final block with feedback parameters is provided which contains:

- The delayed neutron fractions in 6 groups,
- The delayed neutron constants in 6 groups, and
- The inverse Velocities (s/cm) in 7 groups are provided.

5.3.3 Standalone Analysis

The problem setting for the steady-state simulation is based on the C5G7-TD benchmark (Hou et al., n.d.), including the geometry and configuration described previously. The spacer grids are considered in C5G7-TD model (Rubin et al., 2010). The C5G7 pin-wise geometry simulated in NEM is shown in Figure 5-26. The CMR mesh used is a 3×3 coarse mesh, such that each coarse mesh contains 17 fine nodes (i.e., pins) and represents one assembly. On the same figure, each colored pin represents a material, but generally blue colored pins stand for the water reflector while the yellow and green colored portions stand for the UO2 and the MOX assemblies, respectively. The numbers inside each pin cell represent the materials used as explained in Section 5.3.1 and it is shown in Figure 5-26.

The 1D radially collapsed axial power is shown in Figure 5-27. The power profile follows a pure cosine shape and does not see any disturbances as expected for this standalone case with no control rods. Figure 5-28 shows the 2D radial power profile collapsed axially, which again shows the symmetric radial power profile expected from the simulated problem (e.g., in the pins of the MOX assemblies adjacent to the UO2 assemblies). The k-eff computed for this simulation is 1.18659, and we used 1E-05 as a convergence criterion.

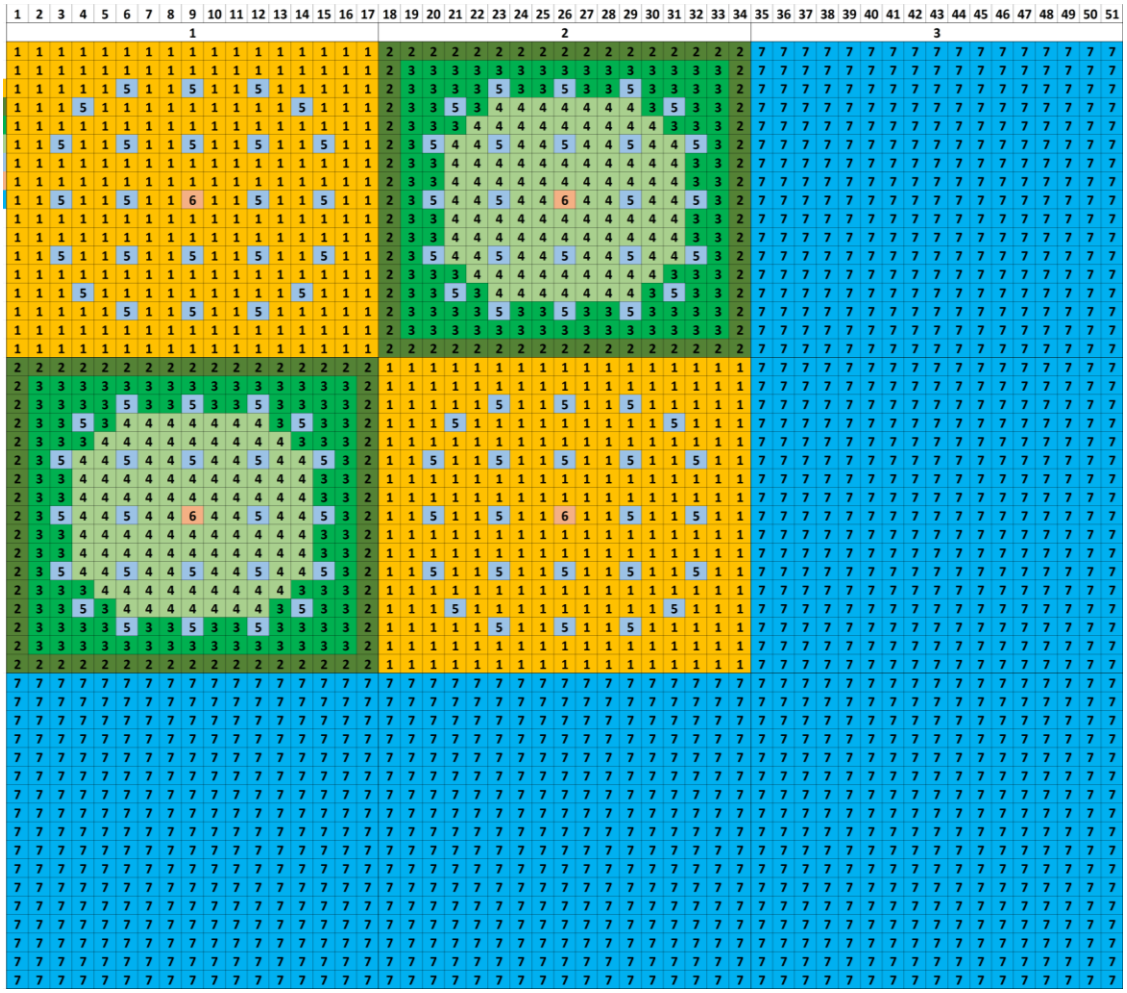


Figure 5-26: C5G7 Pin-Wise Geometry.

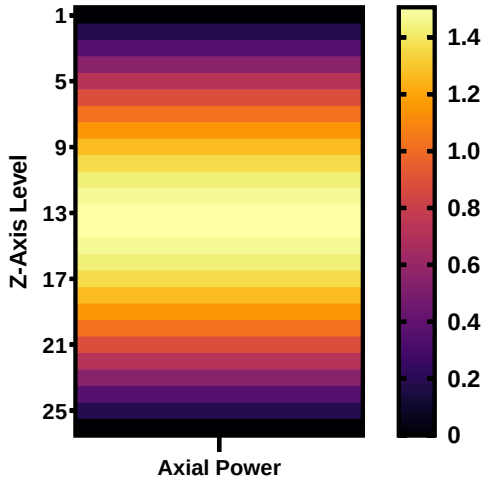


Figure 5-27: The Radially Collapsed 1D Axial Power Profile for Each Z-Axis Level in the C5G7-TD Benchmark.

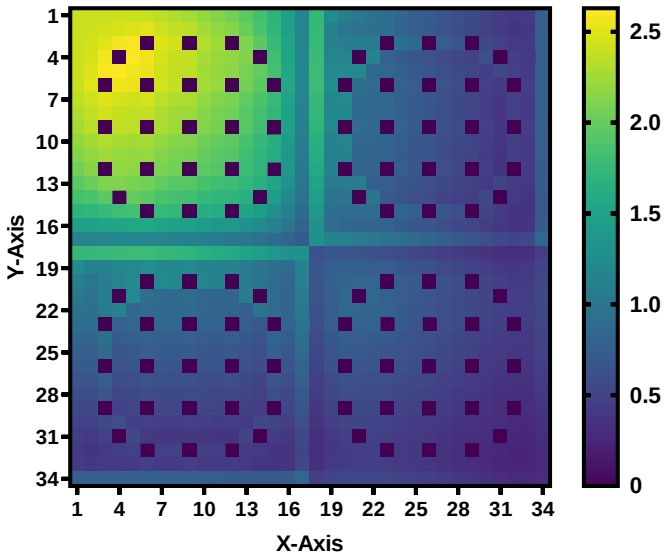


Figure 5-28: The Axially Collapsed 2D Radial Power in Each Pin in the C5G7-TD Benchmark.

5.3.3.1 Performance of the CMR and OpenMP for Pin Wise Geometries

To assess the acceleration configurations and enhancements implemented in NEM-standalone, the C5G7-TD benchmark was chosen since it is a pin-wise configuration that has a set of homogenized cross-sections (cf. Section 5.3.2) that can be used with NEM. This pin-wise configuration proved very demanding for NEM when run on relatively old CPUs (e.g., AMD Opteron Processor), and the runtime taken for a serial run with no acceleration measures was more than 45 minutes. This is expected since the C5G7-TD benchmark has more than 65000 pin-size nodes and 7 energy groups that include up-scattering. Accelerating the NEM computation then seemed obviously a must for such configuration.

Figure 5-29 shows the reduction observed by using the new computational nodes on the RDFMG cluster (based on AMD EPYC 7452). The simulation runtime results are based on an average of 5 simulation runs (except for the old CPU which is based on one simulation). The more than two-fold reduction in the unaccelerated runtime is attributed to the ability of the compiler to optimize better and to use new instructions set not available in the old processors (e.g., AVX-2). The same k-eff and power results were also observed between both old and new CPUs configurations as expected.

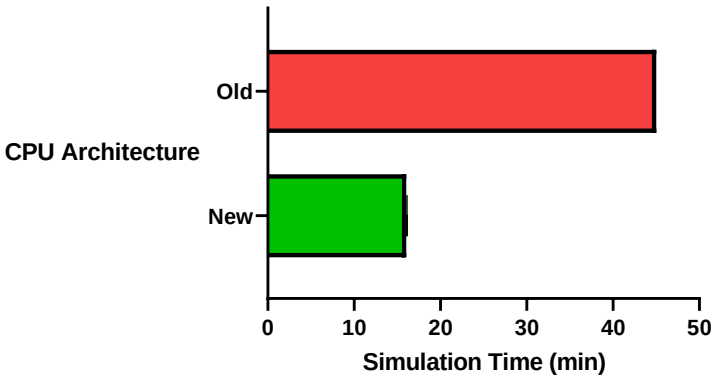


Figure 5-29: Acceleration by Using a Modern CPU Architecture on the New RDFMG Cluster Nodes.

Using CMR and OpenMP has reduced the simulation time further to around 1.5 minutes which translates to more than 10-fold reduction in simulation time when we compare to the unaccelerated simulation. The effect of using several OpenMP threads is shown in Figure 5-30, and no added benefit is seen beyond the 10 threads. This could be attributed to the portions accelerated by OpenMP (as explained in Section 3.2.3) still use serial math libraries (e.g., LAPACK), and hence a math solver library that is also parallelized is desired to increase the benefit of OpenMP implementation.

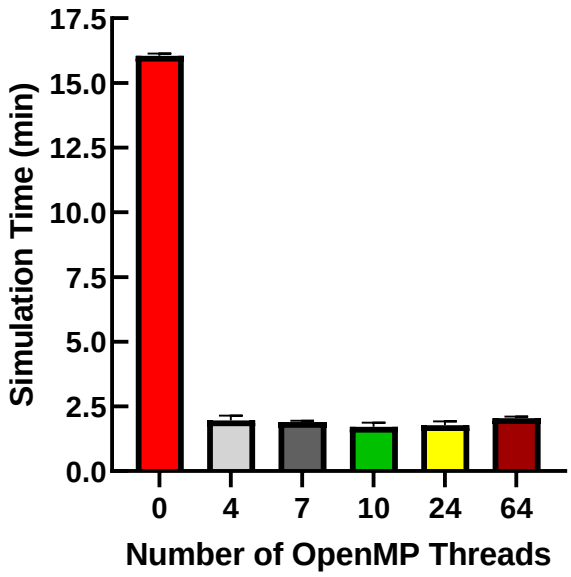


Figure 5-30: Acceleration of the C5G7 Benchmark Simulation by Using Both Several OpenMP Threads and CMR.

A comparison between a serial simulation of the C5G7-TD in NEM, with no measures of acceleration whatsoever, showed different results from the accelerated simulation (e.g., the k-eff difference between both simulations was 14 pcm). The 2D radial power difference between these both simulations are shown in Figure 5-31, and the maximum absolute difference is 0.06%. To isolate the source of this discrepancy, several simulations were carried out while turning the acceleration measures on one at a time. The OpenMP did not show any difference in the simulation results, verifying the implementation process of OpenMP. Using the CMR acceleration procedure showed the mentioned differences, affirming the source of this discrepancy. Inside of the CMR acceleration method, another acceleration measure is used which is the Wielandt-shift power iteration method. As explained in Section 3.2.1, the shift is applied to the eigenvalue to accelerate the computations. Changing either the value of the applied shift or the coarse mesh used (e.g., from the very coarse 3×3 mesh used to a less coarse mesh of 17×17) reduced the differences further.

According to this analysis, the acceleration measures (especially for pin-wise configurations) does show differences to the unaccelerated case (which is expected as explained in Section 3.2), however these differences are insignificant, and the acceleration attained outweigh such drawback.

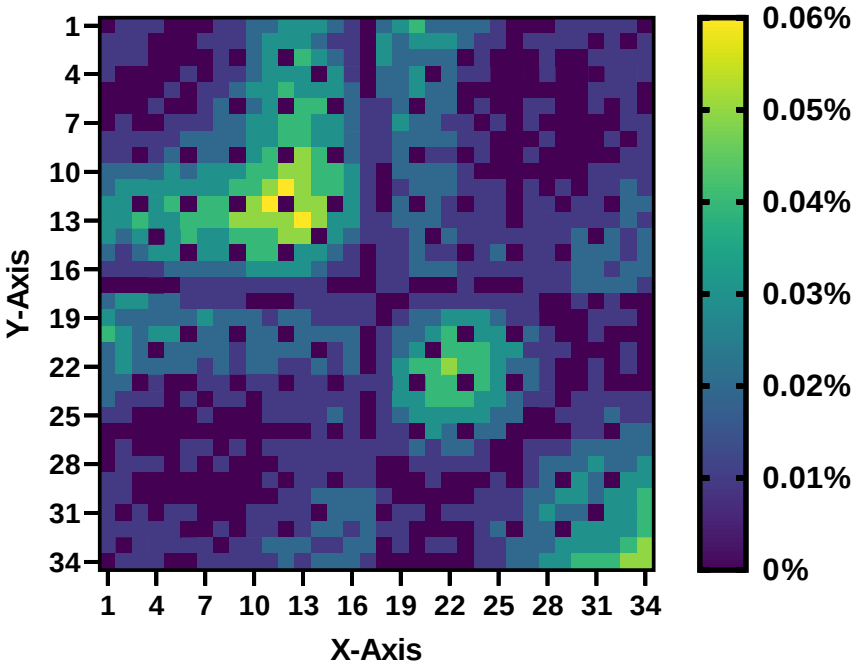


Figure 5-31: The Absolute Difference in 2D Radial Power in Each Pin in the C5G7-TD Benchmark for a Simulation with CMR and with no CMR.

5.3.4 Coupled Analysis

The CTF/NEM results is an important test to verify if the information between the codes is being sending/receiving correctly.

Figure 5-32 presents the radial relative power computed by the CTF/NEM coupled code. Figure 5-33 presents the axial power profile determined by CTF/NEM code. In Figure 5-34 and Figure 5-35 are presented radial and axial fuel coolant outlet temperature distribution respectively. Finally, Figure 5-36 and Figure 5.37 present the radial distribution of core outlet density and the mid-core fuel effective (Doppler) temperature in the C5G7 model.

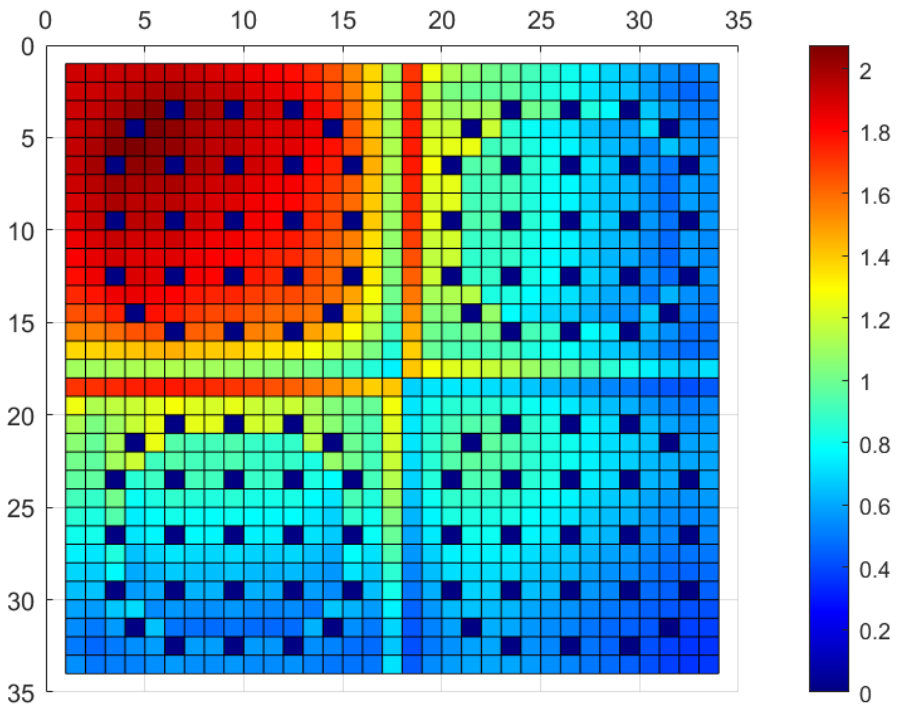


Figure 5-32: CTF/NEM Radial Power Distribution.

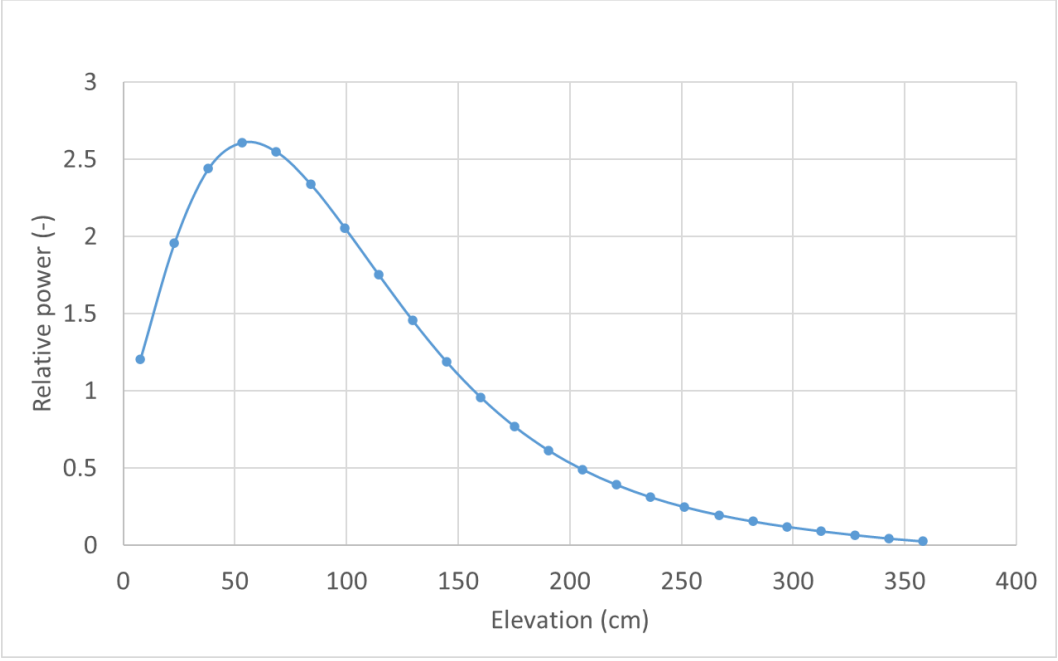


Figure 5-33: CTF/NEM Axial Power Distribution.

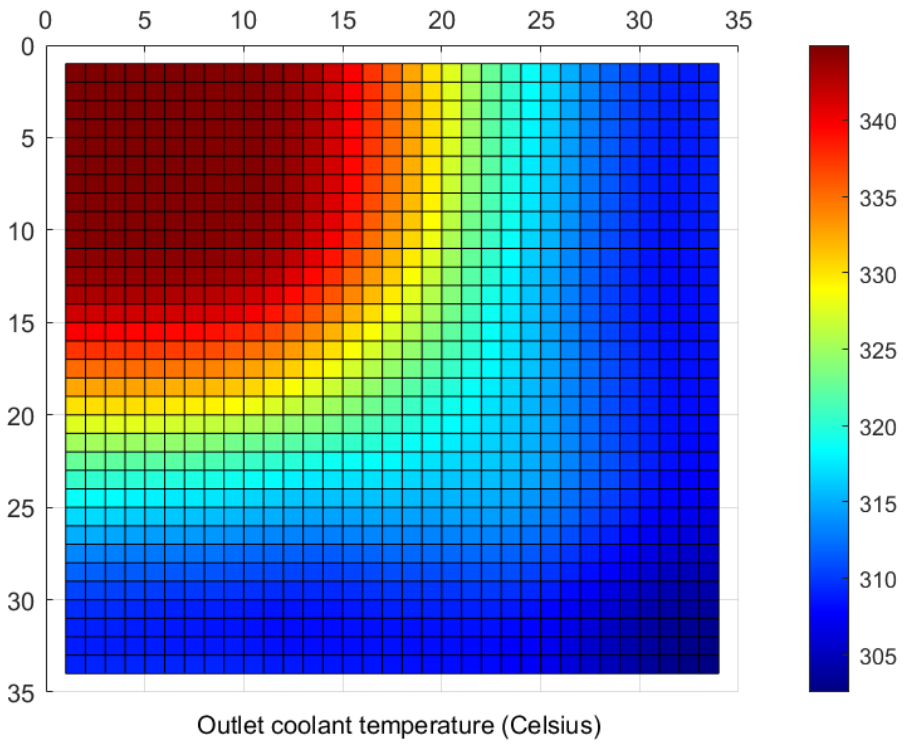


Figure 5-34: Radial Outlet Coolant Temperature Determined by the CTF/NEM Code.

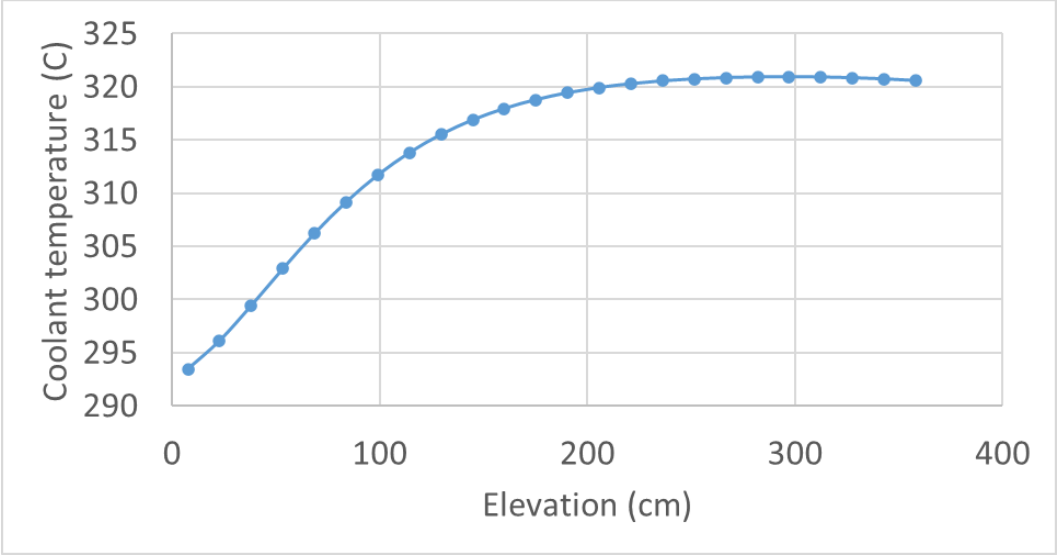


Figure 5-35: CTF/NEM Axial Coolant Temperature.

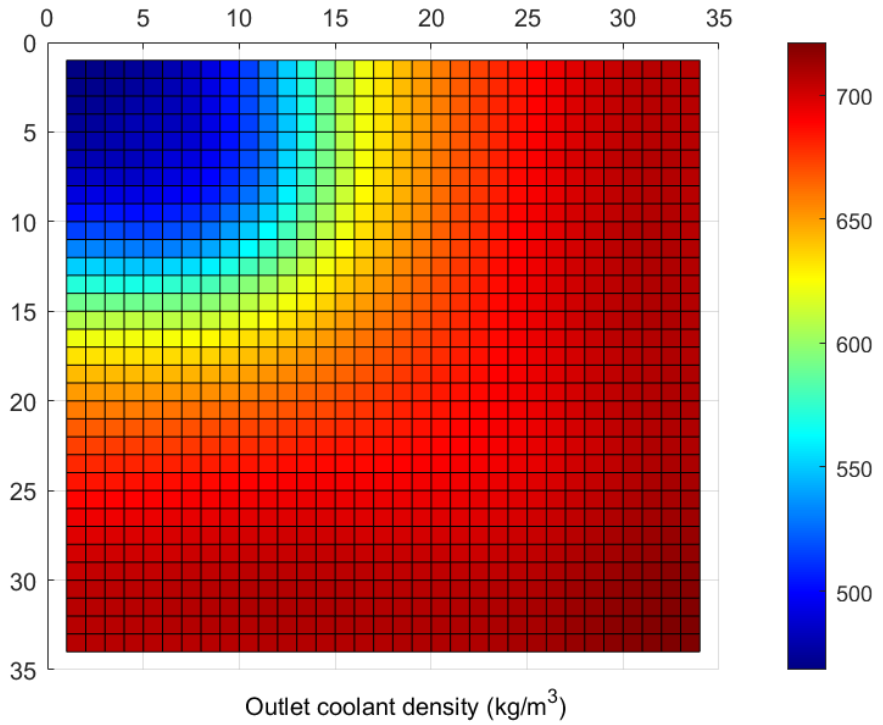


Figure 5-36: Radial Core Outlet Coolant Density Determined by the CTF/NEM Code.

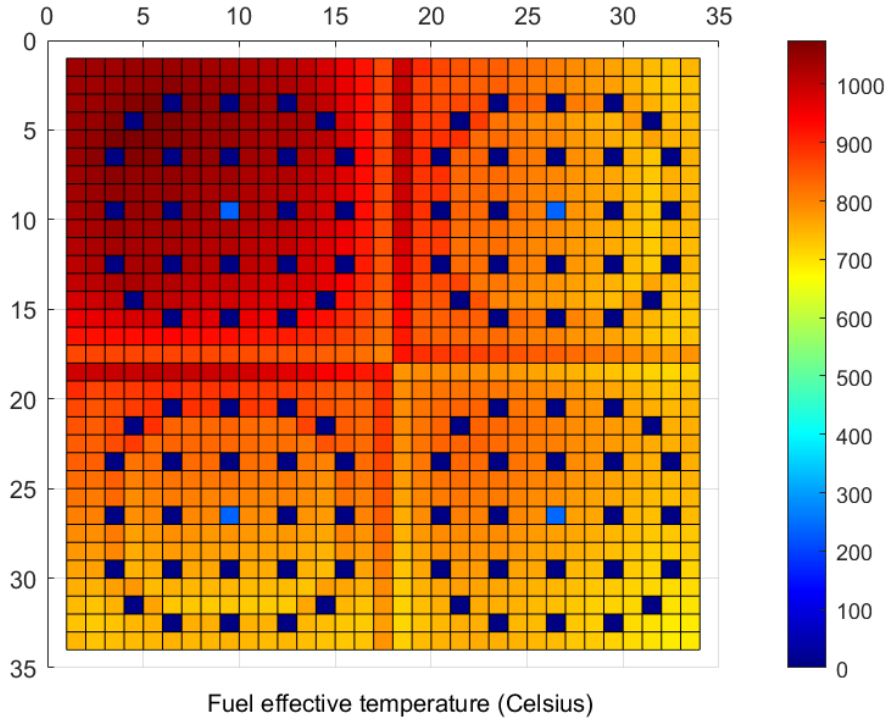


Figure 5-37: Effective Fuel Temperature Determined by the CTF/NEM Code.

5.3.4.1 Performance of the Solution Interruption Method and Balance Method for Pin Wise Geometries

The performance of the solution interruption methods, and the residual balance and adaptive residual balance has not been tested for pin wise diffusion geometries because convergence issues, probably coming from the cross section, are driving to prohibitive running times due to the high number of global iterations needed to converge the case. The analysis of the coupling performance and the coupled steady-state convergence methods have been included as a future line of this thesis. Further improvements of the coupling interface are needed to run the coupled code in parallel, reducing the running time. This will make more plausible the analysis of the performance of the coupled convergence methods as it was done with the nodal case.

CHAPTER 6: CONCLUSIONS AND FUTURE WORKS

6.1 Summary and Conclusions

There are several methods to accelerate the convergence rate of standalone and coupled codes. In this work, we have improved CMR method and proposed and implemented OpenMP method to accelerate the outer iterations in the NEM code. We also implemented two methodologies, Solution Interruption Method, and Residual Balance Method to accelerate the global convergence in the coupled codes.

The contributions of this PhD thesis are:

- Optimization of NEM and stabilization of its asymptotic extrapolation method.
- Improvement of the original CMR method in NEM code.
- Implementation of parallelization (OpenMP) in NEM code to improve the convergence rate of the outer iterations.
- Implementation of the Solution Interruption Method to accelerate the global iteration of the coupled codes.
- Implementation of the Residual Balance Method to improve the convergence of the global iterations of the coupled codes.

To verify the methodologies and the solution obtained, we have used the PWR TMI MSLB Benchmark for the assembly wise spatial resolution case and the C5G7-TD Benchmark for the pin wise spatial resolution case.

It was observed that using CMR with an asymptotic approach could, in certain circumstances, result in divergence, which would mean that one would not be able to reap the acceleration benefits of the methods working together, and modifications were made to the asymptotic extrapolation method and the CMR method. When compared to the earlier version, the

findings of these adjustments demonstrated that the CMR method had been stabilized and made significantly more efficient, resulting in a time savings of approximately 17%.

Still with the objective of improving the performance of the NEM code, using auxiliary tools that allowed a better analysis of the memory used and the frequency of procedure calls within the code, it was implemented parallelization using OpenMP. The performance of the parallelization method was evaluated by simulating the TMI benchmark, and the results showed that CMR reduced the simulation time by two folds. The same behavior was observed for OpenMP using 4 threads without the CMR flag being on, confirming the significance of the implementation of the parallelization method. The acceleration is improved as expected for the parallel case when CMR is turned on, and the average amount of time needed to complete the simulation was 4.237 seconds. For the TMI coupling case, when comparing to the other cases (CMR-off, CMR-on) the OpenMP method presented a better performance during the entire process of calculation presenting a substantial improvement on the NEM code as acceleration mechanism. The maximum relative power distribution difference between CTF/NEM coupling and CTF/NEM PARCS for the TMI model was found to be 0.63%, while the difference found between NEM and PARCS k-eff was around 21.12 pcm.

Both solution interruption and residual balance method were implemented to accelerate the coupled-code simulation convergence. For solution interruption, fewer number of global iterations were needed, and the coupled simulation time was reduced by nearly two-folds (from 29 minutes to ~16 minutes) from a coupled-simulation case with no applied coupled-acceleration. The residual balance outperformed the solution interruption method despite a higher number of global iterations are needed to reach the steady-state convergence. This is due to the fact of each constitutive iteration of CTF and NEM is faster because only a few number of outer iterations are needed to

reach the proposed termination criteria. Must be noted that each individual physics termination criteria depends on the residual of both single physics solutions of the coupled physics simulaion, so the convergence is progressive in both solvers. The time taken for the coupled-simulation of the TMI benchmark was around 10 minutes.

The C5G7-TD, a pin-wise configuration, was particularly demanding for NEM on older CPUs (e.g., AMD Opteron), and a serial run without acceleration, which took more than 45 minutes. When using the new computational nodes on the RDFMG cluster (based on AMD EPYC 7452) the simulation runtime was reduced to less than 20 minutes. Using CMR and OpenMP methods, have decreased simulation duration to 1.5 minutes, a 10-fold reduction from the unaccelerated simulation. A comparison between a serial simulation of the C5G7-TD in NEM, with no measures of acceleration whatsoever, showed different results from the accelerated simulation (e.g., maximum absolute 2D radial power difference between these both simulations was 0.06%). Simulations accelerated with OpenMP alone did not show such differences. The source of this discrepancy was traced to the CMR method and the Wielandt-shift power iteration method. Changing either the value of the applied shift or the coarse mesh used reduced the differences further. The acceleration measures (especially for pin-wise configurations) do show differences to the unaccelerated case (which is expected as explained in Section 3.2), however these differences are insignificant, and the acceleration attained outweigh such drawback.

6.2 Recommendations for Future Work

Concerning the NEM code, other methods such as Multi-Level (ML)-CMR to accelerate the convergence rate could be implemented. The key concept behind multigrid is to speed the convergence of a basic iterative approach by performing periodic global corrections of the fine grid solution approximation, which is performed by solving a coarse one. This recursive approach

is continued until a grid is found where the cost of direct solution is insignificant in comparison to the cost of one relaxation sweep on the fine grid. This multigrid cycle, regardless of the fine grid mesh size, often decreases all error components by a set amount far below one resulting in faster convergence rates. The ML-CMR allows for a significant decrease in the amount of time required to execute industrial whole-core fuel cycle and transient simulations.

Another improvement would be the parallelization of the NEM code using OpenACC. OpenACC is a directive-based C/C++ and FORTRAN programming model and requires significantly less programming effort using low-level model to program heterogeneous high-performance computing hardware architectures. Because OpenACC is a directive-based paradigm that adds clues to a given code base, the code can be built simply in a serial fashion, ignoring the directives, and still produce correct results. Thus, it is possible to conserve or maintain a single code base while providing portability across multiple platforms.

Many of the potential applications that either have associated papers in nuclear literature or could potentially be implemented are still up for consideration. In terms of the applications that involve multi-physics coupling utilizing acceleration techniques cycle depletion and transient calculations will be considered. The ability to do "best-estimate" simulations of interactions is made possible by the capability to evaluate complicated transients at given point of cycle depletion with coupled core-plant interactions. The analysis, and consequently the inclusion of a cross-section library to contemplate the cycle depletion and transient cases will be performed. In order to perform such computations using 3D full-core multi-physics coupled systemsa routine to interpolate in parametrized cross-section libararies will be included to update the cross-sections between each coupled iteration sequence at cycle depletion pint or transient time step

REFERENCES

A. Abarca Giménez, R. Miró, G. Jesús, and V. Martín, "Desarrollo y verificación de una plataforma multifísica de altas prestaciones para Análisis de Seguridad en Ingeniería Nuclear TESIS DOCTORAL," 2017.

A. Abarca, R. Miró, T. Barrachina, and G. Verdú, A PROCEDURE FOR COUPLED THERMAL-HYDRAULIC SUBCHANEL AND NEUTRONIC CODES USING COBRA-TF AND PARCS. 2011.

A. Bennett, N. Martin, A. Bennett, G. Bache, and J. Senecal, "BWR Spacer Grid Modeling Using SERPENT 2 / STAR-CCM+ Coupling." [Online]. Available: https://www.researchgate.net/publication/335988063

A. Ivanov, V. Sanchez, R. Stieglitz, and K. Ivanov, "High fidelity simulation of conventional and innovative LWR with the coupled Monte-Carlo thermal-hydraulic system MCNP-SUBCHANFLOW," Nuclear Engineering and Design, vol. 262, pp. 264-275, 2013, doi: 10.1016/j.nucengdes.2013.05.008.

A. Kiessling, "An Introduction to Parallel Programming with OpenMP," 2009.

A. R. Toth - A theoretical analysis of Anderson acceleration and its application in multiphysics simulation for light-water reactors.

A. Rubin, A. Schoedel, M. Avramova, H. Utsuno, S. Bajorek, and A. Velazquez-Lozada, "OECD/NRC BENCHMARK BASED ON NUPEC PWR SUB-CHANNEL AND BUNDLE TESTS (PSBT) Volume I: Experimental Database and Final Problem Specifications," 2010. [Online]. Available: http://www.nea.fr/html/science/egrsltb/PSBT/.

A. T. Godfrey, "VERA Core Physics Benchmark Progression Problem Specifications," 2014.
A. Toptan, R. K. Salko, M. N. Avramova, K. Clarno, and D. J. Kropaczek, "A new fuel modeling capability, CTFFuel, with a case study on the fuel thermal conductivity degradation." [Online]. Available: http://energy.gov/downloads/doe-public-access-plan

B. R. Bandini, "A THREEDIMENSIONAL TRANSIENT NEUTRONICS ROUTINE FOR THE TRAC-PFl REACTOR THERMAL HYDRAULIC COMPUTER CODE A Thesis in Nuclear Engineering."

Blaise Barney, "Computing at LLNL." https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial (accessed Sep. 09, 2022).

C. Brezinski - Convergence acceleration during the 20th century.

Chandrasekaran S. and Juckeland G., OpenACC for Programmers - Concepts and Strategies. 2018.

D. Gaston, G. Hansen, and C. Newman, "MOOSE: A Parallel Computational Framework for Coupled Systems of Nonlinear Equations 2009 International Conference on Mathematics, Computational Methods & Reactor Physics MOOSE: A parallel computational framework for coupled systems of nonlinear equations," American Nuclear Society, 2009.

D. Padua, Encyclopedia of Parallel Computing.

F. Puente Espel, "HIGH ACCURACY MODELING FOR ADVANCED NUCLEAR REACTOR CORE DESIGNS USING MONTE CARLO BASED COUPLED CALCULATIONS A Dissertation in Nuclear Engineering," 2010.

https://hpc.llnl.gov/software/development-environment-software/intel-vtune-amplifier, "Intel VTune Amplifier."

https://pop-coe.eu/sites/default/files/pop_files/pop-webinar-maqao.pdf, "Guided Performance Analysis and Optimization using MAQAO."

I. Babuska and J. T. Oden, "Verification and validation in computational engineering and science: Basic concepts," Comput Methods Appl Mech Eng, vol. 193, no. 36-38, pp. 4057-4066, Sep. 2004, doi: 10.1016/j.cma.2004.03.002.

I. J. Davis, O. F. Courty, M. N. Avramova, A. T. Motta, and K. N. Ivanov, "HIGH-FIDELITY MULTI-PHYSICS COUPLING FOR PREDICTION OF ANISOTROPIC POWER AND TEMPERATURE DISTRIBUTIONSIN FUEL ROD: IMPACT ON HYDRIDE DISTRIBUTION."

J. Conlin, W. Ji, W. R. Martin, J. L. Conlin, and J. C. Lee, "Pseudo Material Construct for Coupled Neutronic-Thermal-Hydraulic Analysis of VHTGR Monte Carlo Method Development View project Development of fast Monte Carlo code ARCHER View project Pseudo Material Construct for Coupled Neutronic-Thermal-Hydraulic Analysis of VHTGR," 2005. [Online]. Available: https://www.researchgate.net/publication/269997111

J. Hou, K. Ivanov, V. Boyarinov, and P. Fomichenko, "C5G7-TD Benchmark for Time-Dependent Heterogeneous Neutron Transport Calculations." [Online]. Available: https://www.researchgate.net/publication/317663452

J. Neil, "NUCLEAR REACTOR MULTI-PHYSICS SIMULATIONS WITH COUPLED MCNP5 AND STAR-CCM+."

J. P. Senecal and W. Ji, "A MODIFIED TIGHTLY COUPLED METHOD FOR REACTOR TRANSIENT SIMULATIONS."

J. P. Senecal and W. Ji, "Approaches for mitigating over-solving in multiphysics simulations," Int J Numer Methods Eng, vol. 112, no. 6, pp. 503-528, Nov. 2017, doi: 10.1002/nme.5516.

J. P. Senecal and W. Ji, "Development of an efficient tightly coupled method for multiphysics reactor transient analysis," Progress in Nuclear Energy, vol. 103, pp. 33-44, Mar. 2018, doi: 10.1016/j.pnucene.2017.10.012.

J. Wang, X. Li, C. Allison, and J. Hohorst, Nuclear Power Plant Design and Analysis Codes: Development, Validation, and Application: A volume in Woodhead Publishing Series in Energy. Elsevier, 2020. doi: 10.1016/B978-0-12-818190-4.00025-5.

K. Ivanov and M. Avramova, "Challenges in coupled thermal-hydraulics and neutronics simulations for LWR safety analysis," Ann Nucl Energy, vol. 34, no. 6, pp. 501-513, Jun. 2007, doi: 10.1016/j.anucene.2007.02.016.

K. Lipnikov, D. Svyatskiy, and Y. Vassilevski, "Anderson acceleration for nonlinear finite volume scheme for advection-diffusion problems," SIAM Journal on Scientific Computing, vol. 35, no. 2, 2013, doi: 10.1137/120867846.

K. N. Ivanov, T. M. Beam, A. J. Baratta, A. Irani, and N. Trikouros, "PRESSURISED WATER REACTOR MAIN STEAM LINE BREAK (MSLB) BENCHMARK Volume I: Final Specifications US Nuclear Regulatory Commission OECD Nuclear Energy Agency," 1999. [Online]. Available: http://www.copyright.com/.

K. Ni and J. Hou, "An Efficient High-To-Low Informing Scheme for Core Neutronics Calculations Based on NEAMS Tools", doi: 10.13182/PHYSOR22-37331.

Lee D.J. and Downar T.J., "The Application of POSIX Threads and OpenMP to the U.S. NRC Neutron Kinetics Code PARCS."

M. Aldinucci et al., "Practical parallelization of scientific applications with OpenMP, OpenACC and MPI," J Parallel Distrib Comput, vol. 157, pp. 13-29, Nov. 2021, doi: 10.1016/j.jpdc.2021.05.017.

M. Curcic and D. Rouson, "Modern Fortran - Building efficient parallel applications."

M. Hermanns, "Parallel Programming in Fortran 95 using OpenMP," 2002.

M. M. Stulajter, R. M. Caplan, and J. A. Linker, "Can Fortran's 'do concurrent' replace directives for accelerated computing?" Oct. 2021, [Online]. Available: http://arxiv.org/abs/2110.10151

M. Metcalf, J. Reid, and M. Cohen, Modern Fortran Explained.

M. N. Avramova and K. N. Ivanov, "Verification, validation and uncertainty quantification in multi-physics modeling for nuclear reactor design and safety analysis," Progress in Nuclear Energy, vol. 52, no. 7. pp. 601-614, Sep. 2010. doi: 10.1016/j.pnucene.2010.03.009.

OpenACC Programming and Best Practices Guide. https://www.openacc.org/sites/default/files/inline-files/OpenACC_Programming_Guide_0.pdf (accessed Sep. 08, 2022).

P. Birken, "Termination criteria for inexact fixed-point schemes," 2014.

R. S. DEMBOt, S. C. Eisenstat, and T. Steihaug, "INEXACT NEWTON METHODS*." [Online]. Available: https://epubs.siam.org/page/terms

R. Salko et al., "CTF Theory Manual."

R. van Geemert, "MULTILEVEL CRITICALITY COMPUTATIONS IN AREVA NP'S CORE SIMULATION CODE ARTEMIS," American Nuclear Society.

S. P. Kopysov, I. v. Krasnopyorov, and V. N. Rychkov, "CORBA and MPI code coupling," Programming and Computer Software, vol. 32, no. 5, pp. 276-283, Oct. 2006, doi: 10.1134/S0361768806050045.

S. Prema and R. Jehadeesan, "Analysis of Parallelization Techniques and Tools," 2013. [Online]. Available: http://www.irphouse.com/ijict.htm

Senecal J. P., "Efficient Coupling Algoruithms and Reduced-Order Methods for High-Fidelity Multiphysics Simulations of Nuclear Reactors," Rensselaer Polytechnic Institute.

Tammy Carter - Choosing a Parallelization Technique.

The Consortium For Advanced Simulation Of Light Water Reactors_ http://www.casl.gov .

V. Seker, J. W. Thomas, and T. J. Downar, "REACTOR SIMULATION WITH COUPLED MONTE CARLO AND COMPUTATIONAL FLUID DYNAMICS," American Nuclear Society, 2007.

W. Groop, E. Lusk, and T. Rajeev, Using MPI2 - Advanced Features of the Message Passing Interface.

Y. A. Chao, "COARSE MESH FINITE DIFFERENCE METHODS AND APPLICATIONS."

Y. Kozmenkov, S. Kliem, and U. Rohde, "Validation and verification of the coupled neutron kinetic/thermal hydraulic system code DYN3D/ATHLET," Ann Nucl Energy, vol. 84, pp. 153-165, Jul. 2015, doi: 10.1016/j.anucene.2014.12.012.

APPENDICES

Appendix A

A.1 An example of One Cross-Section Set for the C5G7-TD Benchmark

In the following Section, one cross-section set (e.g., the UOX pins set) is given as found in the C5G7-TD benchmark nemtab file. The description of the nemtab file can be found in Section 5.3.2, and the following set follows the general shape described in that Section. The first 10 lines for each cross-section table is given in the following data, and for only one energy group.

```
*

*  NEM-Cross Section Table Input
*
* Number of materials and content of the cross section tables
*  N mat Upsca_XS Xe-XS  Detector-XS  ADFs  Lambdas   Betas
     8       T     F       T         T      T        T

*    Dens mod  T mod      T Fuel          T Clad          Void
     9         5          5
*
*******          X-Section set No.      1
   1
*
*    Group No. 1
*
***************   Diffusion Coefficient  Table
*
5.00000000E+02  6.00000000E+02  7.00000000E+02  8.00000000E+02  9.00000000E+02
1.00000000E+03  1.10000000E+03  1.20000000E+03  1.30000000E+03  3.00000000E+02
4.00000000E+02  5.00000000E+02  6.00000000E+02  7.00000000E+02  6.00000000E+02
6.50000000E+02  7.00000000E+02  7.50000000E+02  8.00000000E+02  1.87709883E+00
1.81438481E+00  1.75591083E+00  1.70124446E+00  1.65003432E+00  1.87707769E+00
1.81436505E+00  1.75589234E+00  1.70122710E+00  1.65002615E+00  1.87705655E+00
1.81434530E+00  1.75587384E+00  1.70120973E+00  1.65000165E+00  1.87702484E+00
1.81431568E+00  1.75584609E+00  1.70119237E+00  1.64998532E+00  1.87704598E+00   …
*
***************   Absorption X-Section Table
*
5.00000000E+02  6.00000000E+02  7.00000000E+02  8.00000000E+02  9.00000000E+02
1.00000000E+03  1.10000000E+03  1.20000000E+03  1.30000000E+03  3.00000000E+02
4.00000000E+02  5.00000000E+02  6.00000000E+02  7.00000000E+02  6.00000000E+02
```

6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 4.91095000E-03
4.93664000E-03 4.96211000E-03 4.98737000E-03 5.01244000E-03 4.91102000E-03
4.93672000E-03 4.96219000E-03 4.98746000E-03 5.01255000E-03 4.91108000E-03
4.93679000E-03 4.96226000E-03 4.98754000E-03 5.01263000E-03 4.91113000E-03
4.93684000E-03 4.96232000E-03 4.98760000E-03 5.01269000E-03 4.91118000E-03
*
********** Fission X-Section Table
*
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 4.24879000E-03
4.25988000E-03 4.27074000E-03 4.28140000E-03 4.29188000E-03 4.24886000E-03
4.25995000E-03 4.27081000E-03 4.28148000E-03 4.29198000E-03 4.24892000E-03
4.26002000E-03 4.27088000E-03 4.28156000E-03 4.29206000E-03 4.24897000E-03
4.26006000E-03 4.27094000E-03 4.28162000E-03 4.29213000E-03 4.24901000E-03
*
********** Nu-Fission X-Section Table
*
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 1.18561000E-02
1.18905000E-02 1.19240000E-02 1.19566000E-02 1.19885000E-02 1.18563000E-02
1.18907000E-02 1.19242000E-02 1.19569000E-02 1.19888000E-02 1.18565000E-02
1.18909000E-02 1.19244000E-02 1.19571000E-02 1.19890000E-02 1.18566000E-02
1.18910000E-02 1.19245000E-02 1.19572000E-02 1.19892000E-02 1.18567000E-02
*
********** Scattering X-Section Table
*
**** From group 1 to 1 *
*** Scattering Order No. 0 *
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 1.11329000E-01
1.14494000E-01 1.17651000E-01 1.20800000E-01 1.23940000E-01 1.11331000E-01
1.14496000E-01 1.17653000E-01 1.20801000E-01 1.23941000E-01 1.11332000E-01
1.14498000E-01 1.17655000E-01 1.20803000E-01 1.23943000E-01 1.11336000E-01
1.14501000E-01 1.17658000E-01 1.20806000E-01 1.23946000E-01 1.11334000E-01
**** From group 1 to 2 *
*** Scattering Order No. 0 *
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 6.12841000E-02

6.42142000E-02 6.71328000E-02 7.00414000E-02 7.29402000E-02 6.12846000E-02
6.42141000E-02 6.71331000E-02 7.00415000E-02 7.29400000E-02 6.12849000E-02
6.42142000E-02 6.71336000E-02 7.00417000E-02 7.29405000E-02 6.12846000E-02
6.42146000E-02 6.71332000E-02 7.00418000E-02 7.29401000E-02 6.12848000E-02
**** From group 1 to 3 *
*** Scattering Order No. 0 *
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 2.54424000E-04
2.73707000E-04 2.92903000E-04 3.12023000E-04 3.31069000E-04 2.54424000E-04
2.73704000E-04 2.92902000E-04 3.12021000E-04 3.31065000E-04 2.54424000E-04
2.73703000E-04 2.92903000E-04 3.12020000E-04 3.31065000E-04 2.54421000E-04
2.73703000E-04 2.92899000E-04 3.12018000E-04 3.31061000E-04 2.54421000E-04
**** From group 1 to 4 *
*** Scattering Order No. 0 *
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 1.26593000E-06
1.36645000E-06 1.46652000E-06 1.56618000E-06 1.66546000E-06 1.26593000E-06
1.36643000E-06 1.46651000E-06 1.56617000E-06 1.66544000E-06 1.26593000E-06
1.36643000E-06 1.46651000E-06 1.56617000E-06 1.66544000E-06 1.26591000E-06
1.36643000E-06 1.46649000E-06 1.56616000E-06 1.66542000E-06 1.26591000E-06
**** From group 1 to 5 *
*** Scattering Order No. 0 *
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 8.86764000E-08
9.57333000E-08 1.02758000E-07 1.09755000E-07 1.16724000E-07 8.86765000E-08
9.57321000E-08 1.02758000E-07 1.09754000E-07 1.16723000E-07 8.86765000E-08
9.57318000E-08 1.02758000E-07 1.09754000E-07 1.16723000E-07 8.86752000E-08
9.57317000E-08 1.02757000E-07 1.09753000E-07 1.16721000E-07 8.86752000E-08 …
**** From group 1 to 6 *
*** Scattering Order No. 0 *
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 6.62705000E-09
7.14477000E-09 7.65953000E-09 8.17176000E-09 8.68163000E-09 6.62711000E-09
7.14465000E-09 7.65952000E-09 8.17169000E-09 8.68150000E-09 6.62713000E-09
7.14463000E-09 7.65958000E-09 8.17169000E-09 8.68155000E-09 6.62702000E-09
7.14470000E-09 7.65942000E-09 8.17168000E-09 8.68139000E-09 6.62702000E-09
**** From group 1 to 7 *
*** Scattering Order No. 0 *

5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 0.00000000E+00
0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
*
********** Assembly Disc. Factor Table - W
*
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
*
********** Assembly Disc. Factor Table - S
*
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
*
********** Detector Flux Ratio Table
*
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02 9.00000000E+02
1.00000000E+03 1.10000000E+03 1.20000000E+03 1.30000000E+03 3.00000000E+02
4.00000000E+02 5.00000000E+02 6.00000000E+02 7.00000000E+02 6.00000000E+02
6.50000000E+02 7.00000000E+02 7.50000000E+02 8.00000000E+02 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
*
********** Detector Microscopic X-Section Table
*
5.00000000E+02 6.00000000E+02 7.00000000E+02 8.00000000E+02
9.00000000E+02 1.00000000E+03 1.10000000E+03 1.20000000E+03

1.30000000E+03 3.00000000E+02 4.00000000E+02 5.00000000E+02
6.00000000E+02 7.00000000E+02 6.00000000E+02 6.50000000E+02
7.00000000E+02 7.50000000E+02 8.00000000E+02 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00
1.00000000E+00 1.00000000E+00 1.00000000E+00 1.00000000E+00 …
*
********** Effective Delayed Neutron Yield in 6 Groups
*
2.17689702E-04 1.46304929E-03 1.34624929E-03 2.79684907E-03 9.32898196E-04
3.16753609E-04
*
********** Decay Constants for Delayed Neutron Groups
*
1.24898324E-02 3.07902044E-02 1.14788227E-01 3.10006284E-01 1.23337044E+00
3.30392133E+00
*
********** Inv. Neutron Velocities for 7 Neutron Groups
*
4.55625671E-10 1.98574053E-09 3.34893164E-08 1.95098316E-07 5.94678040E-07
1.48477027E-06 3.31355076E-06
*