MODULAR OPTIMIZATION CODE PACKAGE: MOZAIK

A Dissertation in
Nuclear Engineering
by
Kursat B. Bekar

© 2008 Kursat B. Bekar

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2008
The dissertation of Kursat B. Bekar was reviewed and approved* by the following:

Yousry Y. Azmy  
Professor of Nuclear Engineering  
Dissertation Co-Advisor, Co-Chair of Committee

Kenan Ünlü  
Professor of Nuclear Engineering  
Dissertation Co-Advisor, Co-Chair of Committee

Jack Brenizer, Jr.  
Professor of Nuclear Engineering  
Chair of the Nuclear Engineering Program

Vijay Narayanan  
Professor of Computer Science

*Signatures are on file in the Graduate School.
Abstract

This dissertation addresses the development of a modular optimization code package, MOZAIK, for geometric shape optimization problems in nuclear engineering applications. MOZAIK’s first mission, determining the optimal shape of the D$_2$O moderator tank for the current and new beam tube configurations for the Penn State Breazeale Reactor’s (PSBR) beam port facility, is used to demonstrate its capabilities and test its performance.

MOZAIK was designed as a modular optimization sequence including three primary independent modules: the initializer, the physics and the optimizer, each having a specific task. By using fixed interface blocks among the modules, the code attains its two most important characteristics: generic form and modularity. The benefit of this modular structure is that the contents of the modules can be switched depending on the requirements of accuracy, computational efficiency, or compatibility with the other modules. Oak Ridge National Laboratory’s discrete ordinates transport code TORT was selected as the transport solver in the physics module of MOZAIK, and two different optimizers, Min-max and Genetic Algorithms (GA), were implemented in the optimizer module of the code package.

A distributed memory parallelism was also applied to MOZAIK via MPI (Message Passing Interface) to execute the physics module concurrently on a number of processors for various states in the same search. Moreover, dynamic scheduling was enabled to enhance load balance among the processors while running MOZAIK’s physics module thus improving the parallel speedup and efficiency. In this way, the total computation time consumed by the physics module is reduced by a factor close to $M$, where $M$ is the number of processors. This capability also encourages the use of MOZAIK for shape optimization problems in nuclear applications because many traditional codes related to radiation transport do not have parallel execution capability.

A set of computational models based on the existing beam port configuration
of the Penn State Breazeale Reactor (PSBR) was designed to test and validate the code package in its entirety, as well as its modules separately. The selected physics code, TORT, and the requisite data such as source distribution, cross-sections, and angular quadratures were comprehensively tested with these computational models. The modular feature and the parallel performance of the code package were also examined using these computational models. Another outcome of these computational models is to provide the necessary background information for determining the optimal shape of the D$_2$O moderator tank for the new beam tube configurations for the PSBR’s beam port facility.

The first mission of the code package was completed successfully by determining the optimal tank shape which was sought for the current beam tube configuration and two new beam tube configurations for the PSBR’s beam port facility. The performance of the new beam tube configurations and the current beam tube configuration were evaluated with the new optimal tank shapes determined by MOZAIK. Furthermore, the performance of the code package with the two different optimization strategies were analyzed showing that while GA is capable of achieving higher thermal beam intensity for a given beam tube setup, Min-max produces an optimal shape that is more amenable to machining and manufacturing.

The optimal D$_2$O moderator tank shape determined by MOZAIK with the current beam port configuration improves the thermal neutron beam intensity at the beam port exit end by 9.5%. Similarly, the new tangential beam port configuration (beam port near the core interface) with the optimal moderator tank shape determined by MOZAIK improves the thermal neutron beam intensity by a factor of 1.4 compared to the existing beam port configuration (with the existing D$_2$O moderator tank). Another new beam port configuration, radial beam tube configuration, with the optimal moderator tank shape increases the thermal neutron beam intensity at the beam tube exit by a factor of 1.8. All these results indicate that MOZAIK is viable and effective and is ready for deployment to address shape optimization problems involving radiation transport in nuclear engineering applications.
# Table of Contents

List of Figures ix  
List of Tables xv  
Acknowledgments xvi  

## Chapter 1  
### Introduction  
1.1 Problem Overview .................................................. 1  
1.2 Background .......................................................... 3  
1.2.1 Shape Optimization ................................................. 3  
1.2.2 Shape Optimization in Nuclear Applications ................. 4  
1.2.3 Introduction to the Neutron Transport Equation ............ 5  
1.2.4 Numerical Solution of the Transport Equation ............... 7  
1.2.4.1 The Multi-group Discrete Ordinates Equations ............ 7  
1.2.4.2 Discrete Ordinates Quadrature Sets ....................... 8  
1.2.4.3 TORT .......................................................... 9  
1.2.5 PSBR Review ....................................................... 9  
1.2.5.1 Core Calculations ........................................... 9  
1.2.5.2 Beam Port Calculations .................................. 11  
1.2.5.3 Experimental Studies ...................................... 14  
1.2.6 Stochastic Methods for Beam Port Calculations ............ 14  
1.2.7 Deterministic Methods for Beam Port Calculations .......... 15  
1.3 Research Objectives and Organization of Study ............... 16  

## Chapter 2  
### Modular Optimization Code Package: MOZAIK  

v
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Introduction</td>
<td>19</td>
</tr>
<tr>
<td>2.2 State-Space Search Representation of the</td>
<td></td>
</tr>
<tr>
<td>Shape Optimization</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Introduction to MOZAIK</td>
<td>20</td>
</tr>
<tr>
<td>2.4 Optimization Sequence</td>
<td>22</td>
</tr>
<tr>
<td>2.5 Organization of Modules</td>
<td>25</td>
</tr>
<tr>
<td>2.5.1 Initializer Module</td>
<td>25</td>
</tr>
<tr>
<td>2.5.2 Physics Module</td>
<td>29</td>
</tr>
<tr>
<td>2.5.3 Optimizer Module</td>
<td>30</td>
</tr>
<tr>
<td>2.5.4 Interface Blocks among the Modules</td>
<td>32</td>
</tr>
<tr>
<td>2.6 Parallel Implementation of MOZAIK</td>
<td>35</td>
</tr>
<tr>
<td>2.6.1 Multi-tasking Module: Parallel Execution of MOZAIK</td>
<td>37</td>
</tr>
<tr>
<td>2.6.2 Organization of Parallel Tasks: Data Decomposition Strategies</td>
<td>40</td>
</tr>
<tr>
<td>2.6.3 Modules Organization for Multi-processor Environment</td>
<td>47</td>
</tr>
<tr>
<td>2.7 Additional Features of MOZAIK</td>
<td>50</td>
</tr>
<tr>
<td>2.7.1 Restart Module: Option “restart”</td>
<td>51</td>
</tr>
<tr>
<td>2.7.2 Debugging Module: Option “debug”</td>
<td>51</td>
</tr>
<tr>
<td>2.7.2.1 Simulate Physics Option</td>
<td>52</td>
</tr>
<tr>
<td>2.7.3 Dual Mesh Module: Option “Dual Mesh Support”</td>
<td>52</td>
</tr>
<tr>
<td>2.8 Physics Code: TORT</td>
<td>53</td>
</tr>
<tr>
<td>2.8.1 Mesh Generation</td>
<td>54</td>
</tr>
<tr>
<td>2.8.1.1 GGTM Code</td>
<td>54</td>
</tr>
<tr>
<td>2.8.2 Cross Section Library Preparation: GIP</td>
<td>55</td>
</tr>
<tr>
<td>2.8.3 Quadrature Selection</td>
<td>55</td>
</tr>
<tr>
<td>2.9 Optimizers</td>
<td>56</td>
</tr>
<tr>
<td>2.9.1 Cost Function</td>
<td>56</td>
</tr>
<tr>
<td>2.9.2 Min-max with Depth-first Search Algorithm</td>
<td>56</td>
</tr>
<tr>
<td>2.9.3 Genetic Algorithms</td>
<td>61</td>
</tr>
</tbody>
</table>

Chapter 3
Model Problem Development: Verification Studies for the Physics Module

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 An Overview of PSBR</td>
<td>64</td>
</tr>
<tr>
<td>3.2 Computational Models</td>
<td>66</td>
</tr>
<tr>
<td>3.2.1 Core Model: MCNP Model of the PSBR and BP-4</td>
<td>67</td>
</tr>
<tr>
<td>3.2.2 Tank Models: TORT and MCNP Models</td>
<td>70</td>
</tr>
<tr>
<td>3.2.3 Beam Tube Models: TORT and MCNP Beam Tube Models</td>
<td>72</td>
</tr>
</tbody>
</table>
3.3 Cross-section Preparation for the TORT Models

3.3.1 ANSL-V Cross-section Library

3.3.1.1 Library Contents

3.3.2 Cross-section Preparation Module for TORT

3.3.3 Sample Cross Section Library for the Model Problems

3.3.4 Testing Generated Cross-section Library with TORT

3.4 Angular Quadrature Selection in TORT

3.4.1 Preparing the Angular Quadrature Set

3.5 TORT Source Module for the Tank Model

3.5.1 MCNP Unit-cell/ADMARC-H Source

3.5.2 Source Module: A Realistic Source Generation Approach

3.5.3 Validation Studies for the Boundary Source

3.6 Streaming Operator

3.6.1 Verification of the Streaming Calculations

3.7 Validation of the TORT Tank Model

3.7.1 Performance of the Models

3.8 Summary of the Computational Models

Chapter 4
Analysis Performed to Test MOZAIK

4.1 Testing the Optimization Sequence

4.1.1 Neutronic Model Development

4.1.2 Preparing the Physics Module

4.1.2.1 Fixing TORT problem parameters

4.1.3 Model Development for Optimization

4.1.4 Optimizer Module

4.1.4.1 Cost Calculations

4.1.4.2 Cost Evaluation

4.1.4.3 New State Generation

4.1.5 A Strategy to Reduce TORT Computation Time

4.1.6 Numerical Results

4.1.6.1 MOZAIK with Min-max.1: Without Streaming Calculation

4.1.6.2 MOZAIK with Min-max.2: With Streaming Calculation

4.2 Performance Evaluation of the Parallel Execution of MOZAIK

4.3 Summary of Testing MOZAIK
Chapter 5
Optimization Calculations with MOZAIK

5.1 First Mission: Determining the Optimal Shape of the D$_2$O Moderator Tank for the Existing Beam Port of PSBR

5.1.1 Neutronic Model

5.1.1.1 Mesh Size Selection

5.1.2 Model Development for the Optimization Calculations

5.1.3 Definition of the Cost Function

5.1.4 Physics Code Preparation

5.1.5 Limiting the Size of the Search Problem Domain

5.1.5.1 Decomposition Methodology

5.1.5.2 Numerical Results

5.1.6 New Source Module: Material Dependent Source Generation

5.1.6.1 New Boundary Source Generation Strategy

5.1.7 Optimization Calculations

5.1.7.1 MOZAIK with Min-max

5.1.7.2 MOZAIK with GA

5.1.8 MOZAIK Results with the Two Optimizers

5.1.9 Smoothing Operation for the Optimal Shape

5.2 New Beam Port Configurations

5.2.1 Tangential Beam Tube Adjacent to the PSBR’s Core Interface

5.2.1.1 Results of MOZAIK with Min-max Optimization

5.2.2 Radial Beam Tube

5.2.2.1 Results of MOZAIK with Min-max

5.3 Summary of MOZAIK’s Performance

Chapter 6
Conclusion and Future Research

6.1 Conclusions

6.2 Future Research

Bibliography
# List of Figures

2.1 Modular Structure ................................................. 21  
2.2 Module Structure in MOZAIK ..................................... 22  
2.3 Original Modular Optimization Procedure .......................... 23  
2.4 Organization of MOZAIK’s modules. ................................. 24  
2.5 Contents of the initializer module .................................. 26  
2.6 Contents of the physics module ..................................... 29  
2.7 Contents of the optimizer module ................................... 31  
2.8 Execution sequence for the serial and parallel versions of the MOZAIK ................................ 38  
2.9 Strategies for distributing tasks in the same optimization iteration; states are identical .......................... 42  
2.10 Strategies for distributing tasks in the same optimization iteration; different states pose different computational loads ..................................... 44  
2.11 Organization of the modules for MOZAIK with the static scheduling algorithm ........................................... 48  
2.12 Organization of the modules for MOZAIK with the self-scheduling algorithm ........................................... 50  
2.13 Search trees for the Min-max algorithm; sample cost function values shown as integers at tree nodes .......................... 57  
2.14 Replacement Operation in Min-max (without pruning) .......................... 59  
2.15 Replacement Operation in Min-max (with pruning) ................. 60  
3.1 Top View of the Existing PSBR Beam Port Layout ..................... 65  
3.2 PSBR beam port locations relative to TRIGA fuel rod’s height ..................... 65  
3.3 Schematic of the full model of the PSBR and beam tubes developed by Sarikaya and Alim .......................... 66  
3.4 Computational models obtained by separating the full neutronics model for BP-4 into three pieces .......................... 67  
3.5 Model geometries generated by gentinp-3 for TORT and MCNP codes (Model at the left side is MCNP5 Core Model, model at the right side is TORT Tank Model) .......................... 69  
3.6 Top-view of the simplified configuration of the beam port of PSBR .......................... 71
3.29 Comparison of partial currents exiting the Bi disk produced by MCNP with two different spatial distributions. 95
3.30 Comparison of partial currents exiting the Bi disk produced by TORT and MCNP with a source with uniform spatial distribution. 96
3.31 Comparison of partial currents exiting the Bi disk produced by TORT and MCNP with a source whose spatial distribution is computed by the source module. 96
3.32 Projection of particle on the exit hole plane. 98
3.33 Beam tube model for validation of the streaming calculation. 99
3.34 Comparison of the outgoing partial currents at the exit hole (computed by the BT models and the streaming operator). 100
3.35 Spatial distribution of neutrons per fission event computed by the streaming operator at the beam tube exit end. 101
3.36 Comparison of the half scalar flux after Bi disk produced by TORT and MCNP Tank Models, and MCNP Core Model. 102

4.1 Top-view of the PSBR’s Tank Models: (a) Tank Model, (b) Tank Model with optimal beam tube location, (c) Simplified Tank Model for the optimization study. 108
4.2 Variation of the neutron partial currents emerging from surface A of the Bi disk for various beam tube locations. 109
4.3 Top and front view of the symmetric model designed for the testing MOZAIK’s performance. 110
4.4 One of the initial state, material distribution (at y=0.0 cm). 113
4.5 Model configuration for computing the cost function via the streaming operator. 115
4.6 Improvement in cost value by Min-max.1 in 87 search iterations. 120
4.7 Optimal D₂O tank shape with Min-max-1. 121
4.8 Top view of the optimal D₂O tank shape with Min-max-1, not showing H₂O. 121
4.9 Back view of the optimal D₂O tank shape with Min-max-1, not showing H₂O. 121
4.10 Front view of the optimal D₂O tank shape with Min-max-1, not showing H₂O. 122
4.11 Comparison of outgoing current spectra after the Bi disk for the real and optimal D₂O tank shapes obtained with Min-max-1. 122
4.12 Comparison of outgoing current spectra at the exit hole for the real and optimal D₂O tank shapes obtained with Min-max-1. 123
4.13 Improvement in cost value by Min-max.2 in 97 search iterations. 124
4.14 Optimal D₂O tank shape with Min-max-2. 125
4.15 Top view of the optimal D$_2$O tank shape with Min-max-2; not showing H$_2$O. 125
4.16 Back view of the optimal D$_2$O tank shape with Min-max-2; not showing H$_2$O. 126
4.17 Front view of the optimal D$_2$O tank shape with Min-max-2; not showing H$_2$O. 126
4.18 Comparison of outgoing current at the exit hole for the real and optimal D$_2$O tank shape with Min-max-2. 127
4.19 Comparison of outgoing current after Bi for the real and optimum D$_2$O tank shape with Min-max-2. 127
4.20 Variation of the total (i.e. cumulative) computation time for the optimization calculation with serial execution of MOZAIK. 129
4.21 Average computation time per state for the optimization calculation with serial execution of MOZAIK. 129
4.22 Variation of the total computation time for the optimization calculation with parallel execution of MOZAIK. 130
4.23 Average computation time per state for the optimization calculation with parallel execution of MOZAIK. 130
4.24 Speedup variation during the optimization procedure (32 processors). 131
4.25 Variation of idleness during the optimization procedure. 132

5.1 CPU Time variation in terms of number of computational cells. 139
5.2 Relative difference between the exit hole neutron partial current computed on a coarse mesh models compared to the value computed with the reference model’s fine mesh. 140
5.3 Sketch of the model problem configuration for the tangential beam tube showing locations of the black absorbers. 142
5.4 Schematic view of the model problem configuration for the radial beam tube. 145
5.5 Comparison of the thermal neutron beam intensities outputs computed by TORT and the streaming operators for both the fixed source approach and the real boundary source. 148
5.6 Comparison of source spectra (normalized to 1 fission event). 150
5.7 Difference between the spectra computed by the source spectrum function obtained by SNNS and the previous source module with the MCNP Core Model. 153
5.8 Difference between the source spatial distributions computed by the source spatial distribution function obtained by SNNS and the previous source module with the MCNP Core Model. 153
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.9</td>
<td>Comparison of the thermal neutron partial current at the exit end of the beam tube computed by TORT and the streaming operator using the SNNS-based source approach and the real source</td>
</tr>
<tr>
<td>5.10</td>
<td>Cost values of the best states as computed by MOZAIK with the two optimizers</td>
</tr>
<tr>
<td>5.11</td>
<td>Thermal neutron intensity of the best states at the exit hole as computed by MOZAIK with two optimizers</td>
</tr>
<tr>
<td>5.12</td>
<td>Number of states in each search iteration for MOZAIK with the two optimizers</td>
</tr>
<tr>
<td>5.13</td>
<td>Average computation time per state in each search iteration for MOZAIK with the two optimizers</td>
</tr>
<tr>
<td>5.14</td>
<td>Front view (from core side) of the optimal shape found by MOZAIK using the Min-max optimization strategy</td>
</tr>
<tr>
<td>5.15</td>
<td>Side view of the optimal shape found by MOZAIK using the Min-max optimization strategy</td>
</tr>
<tr>
<td>5.16</td>
<td>Front view (from core side) of the optimal shape found by MOZAIK using the GA optimization strategy</td>
</tr>
<tr>
<td>5.17</td>
<td>Side view of the optimal shape found by MOZAIK using the GA optimization strategy</td>
</tr>
<tr>
<td>5.18</td>
<td>3D-views of the optimal D$_2$O tank determined by the optimization calculations with MOZAIK (red plane represents the core interface)</td>
</tr>
<tr>
<td>5.19</td>
<td>3D-views of the D$_2$O tank shape smoothed by the shape verifier module (pink shows the smoothed optimal shape)</td>
</tr>
<tr>
<td>5.20</td>
<td>Thermal neutron beam intensity at the exit hole that of the best states computed by MOZAIK with Min-max for the new tangential beam tube configuration</td>
</tr>
<tr>
<td>5.21</td>
<td>Amount of D$_2$O added into search problem domain by Min-max during the optimization procedure</td>
</tr>
<tr>
<td>5.22</td>
<td>Thermal neutron beam intensity of the best states at the exit hole that computed by MOZAIK with Min-max and adaptive source generation approach for the new tangential beam tube configuration, and comparison with values for selected states obtained using TORT with actual boundary source distribution</td>
</tr>
<tr>
<td>5.23</td>
<td>Comparison of the total execution time for MOZAIK with the adaptive (SM-10.0) and SNNS-based (SM-9.0) source generation strategies</td>
</tr>
<tr>
<td>5.24</td>
<td>Speed-up per search iteration of the parallel execution of MOZAIK on 32 processors with the adaptive (SM-10.0) and SNNS-based (SM-9.0) source generation strategies</td>
</tr>
</tbody>
</table>
5.25 Processor idleness per search iteration for the parallel execution of MOZAIK with the adaptive (SM-10.0) and SNNS-based (SM-9.0) source generation strategies ........................................ 171
5.26 3D-views of the optimal D$_2$O tank shape determined by the optimization calculations of MOZAIK with Min-max(before smoothing) 172
5.27 3D-views of the D$_2$O tank optimal shape smoothed by the shape verifier module (pink shows the smoothed optimal shape) ........... 172
5.28 3D-views of the D$_2$O tank optimal shape smoothed by the shape verifier module ................................................................. 173
5.29 Thermal neutron beam intensity at the exit hole of the best state per search iteration as computed by MOZAIK with Min-max for the radial beam tube configuration-I .............................. 175
5.30 Thermal neutron beam intensity at the exit hole of the best state per search iteration as computed by MOZAIK with Min-max for the radial beam tube configuration-II .............................. 175
5.31 3D-views of the optimal D$_2$O tank shape and its smoothed version for the radial beam tube configuration-I ................................ 176
5.32 3D-views of the optimal D$_2$O tank shape and its smoothed version for the radial beam tube configuration-II .............................. 176
## List of Tables

3.1 Neutron energy upper boundaries for the 39-group ANSL-V library (eV) ........................................ 75

3.2 Angular Quadrature sets included in sub-module `xsncon3` .................. 84

4.1 Comparison of TORT calculations with and without the flux initial
guess set to the all H\textsubscript{2}O converged solution ......................... 119

5.1 Several Cartesian meshes employed in determining an efficient mesh
structure for the physics module ..................................................... 138

5.2 Angular flux and its components in the thermal groups computed
by TORT + streaming operator at the beam tube exit hole for the
tangential beam tube for various moderator materials ........................ 146

5.3 Angular flux and its components in the thermal groups computed
by TORT + streaming operator at the beam tube exit hole for the
radial beam tube for various moderator materials ............................. 147

5.4 Number of neutrons per fission event at the boundary source inter-
face for various materials ............................................................. 149

5.5 Outgoing partial current per energy per fission event in the ther-
mal group computed at the beam tube exit end by TORT and the
streaming operator for four beam tube arrangements. ......................... 177
Acknowledgments

I would like to convey my gratitude to all those who made it possible for me to complete this thesis.

I am deeply indebted to my academic advisor Dr. Yousry Y. Azmy for his help, guidance, and stimulating suggestions during all phases of this study. Above all, he provided me with unflinching encouragement and support in various ways.

I would like to record my gratitude to my academic co-advisor Dr. Kenan Ünlü who patiently guided me through the dissertation process.

I gratefully acknowledge all the suggestions of my committee members, Dr. Jack Brenizer and Dr. Vijay Narayanan.

My parents, Selahattin and Şenay Bekar, deserve special mention for their untiring support, and endurance. I would like to give my special thanks to my brother Çağlar Firat Bekar and my sister Ülkü Bekar, and express my sincere appreciation to my friends, Hatice Özlem Özden and Hatice Akkurt for their constant support, encouragement, and patience throughout my PhD program.

Finally, I would like to thank everybody who contributed to the successful completion of this thesis, as well as expressing my apology that I could not mention all the names one by one.
Introduction

1.1 Problem Overview

The neutron beam ports, called also beam tubes, of any research reactor are designed to transport neutrons from reactor core to the reactor bay for experimental needs. The utility of the neutron beam depends on the energy and angular characteristics of the exiting neutron beam, that itself depends on the characteristics of the incoming neutrons at the beam port entry. By performing either stochastic or deterministic neutron transport calculations, these neutron characteristics can be computed for the beam tube (beam port).

The fundamental factors that affect the characteristics of the incoming neutrons to the beam tube are the size and shape of the beam tube, its location and orientation relative to the reactor core, selection of the reflector, moderator and gamma shielding materials, and the neutron energy-angular-spatial distributions at the core surfaces adjacent to the beam tube entrance. The detailed analysis of these components enables performing an optimization calculation to improve the performance of the beam port facility. In addition, necessary information can be obtained in support of designing and installing new experimental devices at the exit end of the beam tube.

The determination of the optimal size and shape of the beam tube design, i.e. the moderator tank, reflector materials, re-entry hole, gamma shielding materials (bismuth, sapphire, lead), collimators, and the beam tube itself enables obtaining a higher neutron output from a beam tube. However, the magnitude of the neutron
output is not the only factor that determines the utility of the neutron beam. Therefore, the optimal size/shape must satisfy certain requirements of the neutron output that depend on the usage of the beam port, such as the beam spatial uniformity desired by experimental users, neutron energy range, and gamma to neutron beam-intensity ratio, while producing higher neutron output.

In order to determine the optimal shape of the components of the beam port design either separately or one at a time, the optimization problem is expressed as a geometric optimization problem. The benefit of this approach is that it generalizes the problem to a broad class for which existing software might be applicable to accomplish certain missions. By constructing a modular code package for this purpose, the contents of various modules can be switched depending on the requirements of accuracy, computational efficiency, or compatibility with other modules. For example a stochastic model can be employed in the particle transport module for the sake of flexible geometric representation of the surroundings, while a deterministic method would be preferable in this module if a detailed distribution of the optimized quantity is required. Similarly, the optimization algorithm itself can be selected from a broad collection of optimization strategies including simulated annealing, genetic algorithms, Min-max, among others.

The Radiation Science and Engineering Center (RSEC) facilities at The Pennsylvania State University (PSU) include the Penn State Breazeale Reactor (PSBR), gamma irradiation facilities, and various radiation detection and measurement laboratories. One of the experimental facilities at the RSEC is the Neutron Beam Laboratory (NBL) including seven beam ports, two of which are available for neutron transmission and neutron radiography experiments. It is not possible to improve the utilization of neutron beams with the current configuration of beam ports and PSBR’s core-moderator assembly. Therefore, additional beam ports to be used with a different experimental setup must be designed around the PSBR core. A study was performed with the support of DOE-INIE funds to investigate new moderator and beam port designs to produce more useful neutron beams.

In this work a new modular optimization code package was implemented and used to obtain an optimal design of the several new beam tube configurations for the PSBR. This illustrates the code package’s success in its first mission and it becomes ready to serve purposes of the shape optimization problems for the other
radiation transport applications in the field of nuclear engineering.

1.2 Background

A detailed literature review was conducted to recognize and benefit from earlier work related to this research.

1.2.1 Shape Optimization

During the last few decades, optimal shape design problems have been actively studied in many engineering applications, such as in computational fluid dynamics applications, structural optimization, component optimization, etc. [1, 2]. The general problem of shape optimization has been the subject of a number of papers [3, 4].

Since the shape optimization problem will be solved numerically, the problem definition must be expressed to the computer by means of programming languages. Therefore, one needs to find a way to represent a shape in discrete form suitable befitting storage in computer memory, and to follow its evolution throughout the optimization procedure.

Several approaches are widely used to solve the shape optimization problem numerically on the computers. One of them is the Lagrangian approach in which the boundary of the shape is followed [3, 4]. The shape boundary is sampled in a relatively dense and uniform manner by considering enough points, then the shape can be evolved by gradually moving the boundary points during the optimization process.

The second approach is the Eulerian approach to construct a special function defined on a rectangular box around the desired shape [4]. This function is chosen to take positive values at points of sought shape, negative values at points outside of it and on the boundary of the shape the function evaluates to zero. After defining this function, it can be evolved by the optimization procedure instead of the shape itself. A rectangular grid structure on the enclosing box helps to sample the function at the grid points. As the shape evolves, the grid points do not change; only the function values at the grid points change, and as they switch
sign this indicates a movement of the shape’s boundary.

A third approach is treating the shape evolution as a flow problem. In this approach, the shape is made of a plastic material which is gradually deformed such that, any point inside or on the boundary of the shape can always be traced back to a point of the original shape in a one-to-one fashion [4].

One of the robust ways for solving the shape optimization problem is to define it as a state-space search [5]. This is a process widely used in the field of artificial intelligence (AI) in which successive configurations or states of an instance are considered and compared with the goal of finding a goal state with a desired property.

In the state space search technique:

- Each one of the possible configurations of the shape is called as a state,
- The set of all possible states for given the problem is called the search space,
- The evaluation or cost function is a function to estimate the value or “goodness” of a state in the search space based on some criteria,
- The objective of the search is to find a goal state that minimizes the cost function over the entire search space.

The state space search approach as used in AI differs from traditional search methods because the state space is implicit: the typical search space is too large to generate and store in memory. Instead, states are generated as they are explored, and typically discarded thereafter. A solution to a combinatorial search instance may consist of the goal state itself, or of a path from some initial state to the goal state.

1.2.2 Shape Optimization in Nuclear Applications

For research reactor applications such as neutron beam optimization, shielding optimization, or cold neutron moderator optimization, most researchers use Monte Carlo codes to model the radiation transport process in order to facilitate definition of complex geometries. The optimization procedures applied are usually manual in nature and limited in scope.
Clark, applied an optimization calculation for designing a cold neutron source [6]. In that study, a Monte Carlo code was written to simulate neutron transport in the defined geometry. The goal of this calculation was to determine a source size and shape that would maximize the cold neutron flux emerging from the source in the direction of the guide tube while keeping the volume and surface area of the source small so as to reduce the heat load. Therefore, different preselected shapes with different sizes were modeled and Monte Carlo simulations were performed. After performing a number of calculations, the best performing shape was identified and considered optimal. The disadvantages of this procedure is that a fixed number of preconceived shapes with various sizes is not representative of the whole search space. Therefore, the identified shape/size might not provide optimal performance. A similar type of calculations was performed for obtaining the optimal shape of the moderator/reflector region feeding neutrons to beam ports for research reactors [7, 8, 9].

Azmy, treats the general shape optimization problem with a novel approach that draws on an analogy with search methods used in artificial intelligence (AI) applications [10]. An AI-type state space search technique for geometric shape optimization was developed and implemented in the modular computer program DAIT. This approach was applied to the problem of optimizing the shape of a liquid-deuterium cold neutron source to maximize its cold neutron output. The major difference of this technique from other techniques applied to the shape optimization for beam ports or cold neutron sources is its automatic operation and the lack of preconceived shapes and sizes to be examined, hence the potential for broader coverage of states comprising the search space. We employ a similar strategy to develop a modular optimization package for the solution of the general class of geometric optimization problems. Since the new code package is illustrated via neutron beam port optimization, the basics of neutron transport theory is introduced to the reader in the following section.

1.2.3 Introduction to the Neutron Transport Equation

The neutron transport equation is given by the linear form of the Boltzmann equation. The linear form is derived by ignoring neutron-neutron interactions.
The time independent neutron transport equation is given by [12],

\[
\hat{\Omega} \cdot \nabla \Psi(\vec{r}, E, \hat{\Omega}) + \sigma_t(\vec{r}, E)\Psi(\vec{r}, E, \hat{\Omega}) = \int_0^\infty dE' \int_{4\pi} d\hat{\Omega}' \sigma_s(\vec{r}, E' \to E, \hat{\Omega}', \hat{\Omega})
\]

\[
\Psi(\vec{r}, E', \hat{\Omega}') + \frac{\chi(E)}{4\pi} \int_0^\infty dE' \nu \sigma_f(\vec{r}, E') \int_{4\pi} d\hat{\Omega}' \Psi(\vec{r}, E') + \hat{S}(\vec{r}, E, \hat{\Omega})
\]  

(1.1)

In this equation:

- \(\vec{r}\) is the position vector of the particle in physical space,
- \(E\) is the energy of the particle,
- \(\hat{\Omega}\) is the unit vector along the direction of the particle’s motion,
- \(\Psi(\vec{r}, E, \hat{\Omega})\) is the particle angular flux,
- \(\sigma_t(\vec{r}, E)\) is the total interaction macroscopic cross section at position \(\vec{r}\) and energy \(E\),
- \(\sigma_s(\vec{r}, E' \to E, \hat{\Omega}', \hat{\Omega})\) is the macroscopic double differential scattering cross section,
- \(\chi(E)\) is the fraction of the fission neutrons emitted per unit energy,
- \(\nu\) is the average number of neutrons emitted per fission event,
- \(\sigma_f(\vec{r}, E')\) is the macroscopic fission cross section,
- \(\hat{S}(\vec{r}, E, \hat{\Omega})\) represents the fixed distributed source, if any,
- \(\hat{\Omega} \cdot \nabla \Psi(\vec{r}, E, \hat{\Omega})\) is the streaming term, it represents the flow of neutrons,
- \(\sigma_t(\vec{r}, E)\Psi(\vec{r}, E, \hat{\Omega})\) is the collision term, it represents the removal rate of neutrons due to all reaction types,
- \(\int_0^\infty dE' \int_{4\pi} d\hat{\Omega}' \sigma_s(\vec{r}, E' \to E, \hat{\Omega}', \hat{\Omega})\) is the in-scattering term, it represents the rate scattering of particles,
\[
\chi(E) \frac{\nu \sigma_f(\vec{r}, E')}{4\pi} \int_0^\infty dE' \nu \sigma_f(\vec{r}, E') \int_{4\pi} d\hat{\Omega} \Psi(\vec{r}, E') \text{ is the fission term, it represents the rate of fission neutrons generated,}
\]

The terms on the left hand side of the Eq. 1.1 represent neutron loss mechanisms while the terms on the right hand side of the Eq. 1.1 represent the production of particles. This equation can be rewritten by introducing a general source term \( \hat{q}(\vec{r}, E, \hat{\Omega}) \) which includes the scattering term, the fission term and the external source term

\[
\hat{\Omega}.\nabla \Psi(\vec{r}, E, \hat{\Omega}) + \sigma_t(\vec{r}, E) \Psi(\vec{r}, E, \hat{\Omega}) = \hat{q}(\vec{r}, E, \hat{\Omega}) \tag{1.2}
\]

Evidently \( \hat{q}(\vec{r}, E, \hat{\Omega}) \) depends on the solution of Eq. 1.2, \( \Psi(\vec{r}, E, \hat{\Omega}) \), which itself depends on \( \hat{q} \) thus calling for an iterative solution algorithm.

### 1.2.4 Numerical Solution of the Transport Equation

In order to solve the transport equation with a deterministic method, discretization of the spatial, energy and angular variables is applied to the transport equation (Eq. 1.2). By dividing the energy range into intervals, or groups, and integrating Eq. 1.2 over each energy interval \([E_g, E_{g-1}]\), energy discretization is achieved and a multi-group form of the transport equation is obtained,

\[
\hat{\Omega}.\nabla \Psi_g(\vec{r}, \hat{\Omega}) + \sigma_{tg}(\vec{r}) \Psi_g(\vec{r}, \hat{\Omega}) = \hat{q}_g(\vec{r}, \hat{\Omega}) \tag{1.3}
\]

#### 1.2.4.1 The Multi-group Discrete Ordinates Equations

The Discrete Ordinates Method (Sn) is one of the most widely used techniques to solve the neutron transport equation in terms of discretization of the angular variable. In this method, the Boltzmann equation is solved along a finite number of discrete directions \( \hat{\Omega}^m \), to each of which is associated a weight \( \omega^m \). Each weight represents an area \( \Delta \hat{\Omega}^m \) on the unit directional sphere. The multi-group neutron transport equation for arbitrary direction \( \hat{\Omega}^m \) is given by,

\[
\hat{\Omega}^m . \nabla \Psi_g^m(\vec{r}) + \sigma_{tg}(\vec{r}) \Psi_g^m(\vec{r}) = \hat{q}_g^m(\vec{r}) \tag{1.4}
\]

In an analogous way, the spatial variable(s) is defined in discrete form in terms of a suitable orthogonal coordinate system (Cartesian, Cylindrical or Spherical).
for Eq. 1.4, and then a mesh-sweep procedure is applied using different flux evaluation models to solve for the cell-averaged angular flux. Inner and outer iterations are used to reconcile the scattering and fission sources constituting \( \hat{q} \) with the computed cell-averaged angular flux and its angular moments.

### 1.2.4.2 Discrete Ordinates Quadrature Sets

There are several techniques for the generation of discrete ordinates and associated weights [13, 12]. Level symmetric quadratures (LSQ) are most widely used for general applications. Full symmetry requires that the set of angles \( \hat{\Omega} \), be invariant under all 90 degree rotations about the three coordinate axes. Therefore, each set of ordinates must be symmetric with respect to the origin and at the same time the set of points on each axis must be the same. For N levels, this constraints produces a total of N(N+2) directions is obtained on the unit sphere.

The usage of higher order angular quadratures increases the solution’s detail at smaller surface at which the flux is calculated and reduces undesirable ray-effects in weakly scattering configurations. On the other hand, it’s usage increases TORT’s computational time since the latter grows with the number of angles employed in the angular quadrature set employed. Therefore, some biased quadratures were introduced to obtain detailed angular behavior near, say, the z-axis, in reasonably shorter execution times [13].

Biased quadrature sets do not have an equal number of directions in the positive and negative domain of one of the variables. In other words, they are “biased” by having a larger number of directions in some portion of the unit sphere. Therefore, the biased quadrature sets are usually used when the neutron flow is highly anisotropic in some preferred directions. The directions in the unbiased hemisphere are taken from half of a fully symmetric set. e.g. \( S_{10} \). In contrast the directions in the biased hemisphere is from half of the fully symmetric set in which the first \( \eta \)-level has been replaced by a set of new levels [13]. In this way the number of directions in biased hemisphere is increased while the number of directions in the unbiased hemisphere remains constant.
1.2.4.3 TORT

TORT is a 3-D discrete ordinates code that is suitable for cylindrical (RΩZ) or Cartesian (XYZ) geometry, as well as several two-dimensional coordinate systems [14]. It calculates the neutron flux and/or photons throughout three-dimensional systems due to particles incident upon the system’s external boundaries, due to fixed internal sources, or due to sources of secondary particles generated by interaction with the system materials. The linear Boltzmann transport equation is solved using the method of discrete ordinates to treat the directional variable. The weighted difference, nodal, or characteristic methods are available to treat spatial variables. Energy dependence is treated using a multi-group formulation. Anisotropic scattering is treated using a Legendre expansion of the scattering kernel and spherical harmonics expansion of the angular flux, and the latter expansion is mapped to the discrete ordinates representation of the flux. Two levels of iterations, inner and outer, are used to resolve implicitness of the angular flux’s solution caused by scattering between directions within a single energy group, by scattering from one-energy group to another group, by fission, and by certain types of boundary conditions. Methods are available to accelerate convergence. Fixed sources can be specified at either external or internal mesh boundaries, or distributed within mesh cells.

1.2.5 PSBR Review

1.2.5.1 Core Calculations

The PSBR is a light water cooled, pool type reactor with natural circulation cooling. MTR type fuel elements were used in its original core for 200 kW(t) steady-state power. In 1965, its core and control system were replaced by an advanced General Atomics TRIGA core which is capable of operating at a 1MW(t) steady-state power level with pulsing up to 2000 MW. It has been used since then for experimental and educational purposes.

The PSBR’s TRIGA core was first loaded in 1965 with only 8.5% wt $ZrH_x-U$ fuel. It has been reloaded with fresh 12% wt $ZrH_x-U$ fuel elements (six elements at each reload) since 1972. Currently, it is operated using core cycle 52. In the past, the core fuel management plan of PSBR has been developed and verified based on
TRICOM, the TRIGA core management model, which was developed based on the codes, PSU-LEOPARD, EXTERMINATOR2, and MCRAC [15]. Although this model was used for many years, large uncertainties are introduced in the calculated parameters by the modeling limitation of these outdated computational tools. Thus, a normalization had to be performed between the calculated data and the measured data after each calculation to improve the accuracy of the results.

Y. Z. Kim, developed analytical models of the TRIGA core configuration based on the stochastic solution of the neutron transport equation (Monte Carlo Method) in his MS thesis [16]. The core power distribution is computed by using the MCNP code for criticality calculations and ORIGEN2 for the depletion calculations [17, 18]. In the analyzed core configuration, even though the core power peaking factor was extremely high, it was possible to reduce it by determining a proper core configuration such as shuffling the fuel elements to lower the peaking factor. Although a better configuration can be obtained via that procedure, the required Monte Carlo based calculation is not practical for repeated use while the best configuration is sought because of the extremely long computation time consumed by MCNP. However, Monte Carlo computed results can be used as reference values to verify other, more efficient, calculations with deterministic solution methods.

N. Kriangchairporn, developed a new Advanced Fuel Management System based on the HELIOS lattice-physics code and the multi-dimensional nodal diffusion code ADMARC-H in her MS thesis [19, 20]. The resulting computational procedure eliminated the deficiencies of the outdated TRICOM code system on both levels: the cross-section generation and the core simulation. In this model, HELIOS improved the geometry modeling by explicitly modeling the hexagonal unit cell, and by performing the calculation separately in each region, i.e. fuel and moderator, allowing a better thermalization model. The ADMARC-H code enables a 3-D full core hexagonal geometry model and a 3-D macroscopic semi-implicit burnup model [21]. In this model, the whole core calculation was performed by using the diffusion approximation with 2-D cross-section generation and depletion. The flux distribution computed by this code system can be used to drive the beam port calculations performed in the process of redesigning the PSBR beam port facilities, and for the comparison and verification of transport methods’ solutions.
B. Sarikaya, developed a beam port analysis code system (PSUBACS) for PSBR beam port calculations [22]. In this code system, the core calculations were performed using the nodal diffusion code ADMARC-H. The few-group cross section library needed to perform the diffusion calculations with ADMARC-H was generated by the lattice physics code HELIOS. This diffusion core calculation was performed to obtain a boundary source for beam port transport calculations that were subsequently performed by MCNP. The PSBR core cycle 51 was modeled in full 3-D geometry using ADMARC-H, which calculates the eigenvalue, the normalized power distribution, and two-group flux and partial current distributions based on the cross section library generated by HELIOS. This study utilized the two-group incoming partial current output option available in ADMARC-H since a directional source distribution was needed for the MCNP calculations.

A detailed Monte Carlo calculation was performed on the PSBR core for depletion calculations [23, 24]. In this study, a preliminary calculation was employed for PSBR fuel elements. Moreover, a code system, TRIGSIM, was developed to generate MCNP and ORIGEN models for the PSBR automatically. This MCNP core model is useful to obtain a boundary source for the calculation of existing beam ports and designing new beam ports. In addition, its result can be used to verify the boundary source calculation from PSUBACS.

1.2.5.2 Beam Port Calculations

The PSBR has seven beam ports for experimental purposes. They were designed for the original PSBR core. After replacing its core with TRIGA type fuel, an elevation difference occurred between the centerline of the new core and six of the beam tubes because the height of the new element (15 inches) is shorter than the height of the original MTR fuel elements (24 inches). Therefore only one beam port BP#4 is in the centerline of the TRIGA fuel element.

Currently, the existing D$_2$O moderator tank and graphite reflector block design enables the use of two beam ports, BP#4 and BP#7, for neutron radiography and neutron transmission measurements [25]. The remaining 5 tubes are not used because the current moderator tank design only supports two beam ports and with the TRIGA core arrangement they point below the extent of the core. When the reactor core is placed next to a D$_2$O tank and graphite reflector assembly
located at the entering tip of the beam port (BP) locations, thermal neutron beams become available for neutron transmission experiments and neutron radiography measurement from beam ports BP#7 and BP#4.

Initially, the D$_2$O tank was designed in the shape of a drum whose diameter was 34 inches and height was 27 inches. A. Haghighat, C. Pozsgai and J. Wagner developed a 3-D, continuous energy, Monte Carlo model to determine the optimal size of the D$_2$O tank and compute the neutron and gamma flux along the neutron collimator [26]. In the model, a plane neutron source whose characteristics (spatial, energy and angular distributions) were not reported was placed at the interface between the PSBR core and the D$_2$O tank to represent the reactor. Thus, the model includes only the D$_2$O tank and the collimators. A couple of different tank sizes were modeled and their results were compared. In this preliminary report, it was stated that the size of the D$_2$O tank was too large and designing a new collimator was proposed [26].

Subsequently, J. Wagner and A. Haghighat modified the Monte Carlo model for the optimization of the PSBR D$_2$O Tank [27]. In this study, the spectrum of the source neutrons was set to a Watt fission spectrum, and the cross-sections were upgraded from ENDF/B-V to ENDF/B-VI. Moreover, the source was changed from a rather coarse array of discrete points to a surface source over the face of the core and a Boral shield was added to the back of the D$_2$O tank. At the same time, a gamma source was introduced at the same surface as the neutron source having a typical gamma spectrum from a fission reactor. In order to obtain the optimal size of the D$_2$O tank, several sizes and configurations were examined and compared to the reference tank dimension (D=34 inches, H=27 inches) by performing both gamma and neutron calculations on the considered models. The results of this study show that the optimal size should be smaller than the reference model, but no statement is made specifying the optimal size of the D$_2$O tank.

In addition, employing a fission spectrum for the neutron source at the core interface is inaccurate because the PSBR hydride fuel elements give rise to a softer spectrum at that interface.

B. Sarikaya et al., performed some calculations for the beam port BP#4 [22]. In order to obtain reasonable results, the full model was divided in two parts: a D$_2$O tank model and a beam tube model. For the D$_2$O tank model, a boundary
source was computed by using the spatial distribution from an ADMARC-H core calculation and the energy distribution from an MCNP core calculation. A uniform angular distribution for this surface neutron source was selected since ADMARC-H’s diffusion model does not provide angular detail for the neutron flux and current. The neutron intensity of the outgoing collimated beam was then computed by enabling the directional tally option in MCNP. Although the directional tally option was enabled for the MCNP calculations to compute the intensity of the outgoing neutron beam (highly collimated), only the half-scalar flux on the exit face of the Bi disk (place at the beam tube entrance for gamma shielding purposes) was possible to compute due to the poor statistics in directional tally results.

This neutron source (tallies) at the exit edge of the Bi disk was used to compute the boundary source for the beam tube model. By using these approximations, the results obtained at the beam tube exit had a 14% overall difference between the computed beam intensity and experimentally measured values. In addition, a simple optimization calculation was performed on BP#4. Due to MCNP’s long computation time, this optimization was performed manually by selecting a couple of beam tube locations. By changing the beam tube location in the D$_2$O tank both horizontally and vertically, the best location for the beam tube was determined from among the tested positions. In this calculation, the cost value was set to the total number of outgoing neutrons at the exit surface of the Bi gamma shielding disk. The results show that the neutron beam intensity increases if the beam tube gets closer to the reactor core interface.

B. Sarikaya, and F. Alim performed detailed calculations for the beam port facility of the PSBR [28, 29, 30]. Two different MCNP models were introduced: MCNP full model including the PSBR core, the D$_2$O tank and a portion of the beam tube, and MCNP D$_2$O tank model including the D$_2$O tank and a portion of the beam tube. In order to simulate core neutrons incoming into the D$_2$O tank model, a boundary source was computed at the core tank interface by performing ADMARC-H and MCNP calculations. Results of a simple manual optimization were also given. In addition, some calculations were performed to identify the origin of the gamma photons arriving at the beam tube exit (from the core or from $(n, \gamma)$ reactions in the reactor pool and beam port materials).
1.2.5.3 Experimental Studies

J. Niederhaus, prepared an experimental setup to measure the thermal neutron flux output of the beam ports of PSBR [31, 32, 33]. A single disk, slow chopper system was used and the primary beam at PSU RSEC was characterized with it. By performing time-of-flight spectroscopy across a known distance, the neutron energy distribution was obtained and recorded. The recorded data was transformed to yield the speed and energy spectrum of neutrons. This data was compared with models based on the Maxwell-Boltzmann distribution. This comparison facilitated characterization of the neutron beam. These results are useful for comparing, verifying and validating computational models of the beam ports of the PSBR.

1.2.6 Stochastic Methods for Beam Port Calculations

Monte Carlo codes such as MCNP [17] are usually preferred for neutronic calculations because of their ability to accommodate complex geometries. The Monte Carlo method is based on a stochastic process; a particle is simulated from its birth (generation after a certain reaction or directly from a fixed source) to its death (absorption within the system or escaping out the boundaries) by defining a probability distribution for each event it encounters. Therefore, in order to compute some quantity at a specified location/surface, independent particle simulations should be performed many times. The statistical error decreases as the number of histories increases. The disadvantage of this method is its long computation time necessary to obtain reasonably accurate results at the end of the simulation. For this reason, it is not useful in the state-space search algorithm that is the ultimate target of the project. In addition, analog MCNP techniques are not suitable for deep penetration problems. Therefore, typically this type of calculation is performed using biased techniques to efficiently reduce the statistical error in the computed quantities. Although Monte Carlo may be used in the optimization procedure with the support of with some biasing techniques, this still is not a reasonable approach because most of the biasing techniques are material and configuration dependent. As explained later, material composition of the beam port surroundings is changed from one state to another during the search process.

The beam port calculations require a very detailed geometrical description of
the system. This complexity of the geometrical model and the number of materials used encourages researchers and engineers to perform MCNP calculations for beam port modeling, beam optimization, shielding optimization and moderator optimization.

Tetsuo Matsumoto performed some MCNP calculations for a TRIGA reactor [34, 35]. In the calculations, several prospective filters: high-density graphite (for thermal neutron beam generation), bismuth (for gamma ray shielding), aluminum, aluminum oxide, aluminum fluoride (for epithermal neutron generation) and lead fluoride (for gamma ray and fast neutron shielding) were examined for obtaining an intense thermal neutron beam for various applications. With a suitable filter arrangement, thermal and epithermal neutron beams were computed as $2 \times 10^9$ and $7 \times 10^8$ n/cm$^2$-s, respectively.

Kalpakkam Mini Reactor (KAMINI) was modeled by using MCNP with continuous energy cross-sections [37, 38]. The measured axial neutron flux profile and foil reaction rates in one of the in-core irradiation locations and foil reaction rates at the beam port were compared with the computed results. The computed results conformed with the experimental measurements. In addition, an optimization calculation was performed for the shield structure for two of the beam tubes one towards the west the other towards the south sides of the reactor. The performed optimization was not automatic. A number of configurations were prepared, the computational model solved for each, then the computed results were compared and the best performer selected to represent the optimal configuration.

1.2.7 Deterministic Methods for Beam Port Calculations

Deterministic transport methods can be used for the same purposes as Monte Carlo methods. The main advantage of these methods is that usually the detailed solution distribution (in space, angle and energy) may be obtained in a shorter computational time.

One of the deterministic solvers used in the proposed optimization algorithm is TORT [14]. TORT is a 3-D discrete ordinates neutral particle transport code that is suitable for Cartesian and cylindrical geometry. The linearized Boltzmann transport equation is solved in TORT using the method of discrete ordinates to
treat the directional variable. In order to treat the spatial variables, weighted difference, linear nodal and linear characteristics methods are available in TORT. The energy dependence is treated using the multi-group formulation. By using a Legendre expansion, anisotropic scattering is represented to high orders. TORT can perform both adjoint and forward, fixed source and eigenvalue calculations.

In the literature, TORT was used for dose/flux calculation in large scale problems [39, 40], response calculation for variance reduction of Monte Carlo calculations [41], shielding calculations [42], and fluence calculations [43].

Usually deterministic codes, like TORT, are preferred for problems in which the detailed distribution of the computed quantity is required, or for large scale problems that cannot be simulated by stochastic codes without biasing. TORT is also preferable for problems where the results must be computed with reasonable accuracy in a short time interval.

### 1.3 Research Objectives and Organization of Study

The objective of this study was to develop a modular optimization code package (MOZAIK) to address geometric optimization problems in nuclear related applications, and to employ this code package for designing new beam ports for PSBR.

Initially, a neutronic model was developed for one of the existing beam ports of PSBR to obtain the necessary background information to use in designing new beam ports with their moderator/reflector assembly around the reactor core. The objective of this preliminary step was to test and validate the selected 3-D deterministic transport solver, TORT, and the requisite data such as source distribution, multi-group cross-section library and angular quadratures for the existing beam tube calculation. In order to verify TORT’s results, the same configurations were simulated by performing MCNP calculations. Furthermore, a streaming operator was developed and implemented to compute the neutron beam spectrum and spatial distribution at the beam tube exit without performing a full transport simulation along the length of the beam tube.

In the second stage, the optimization sequence was constructed by developing
the modules of the modular optimization code package. In order to test the modular feature of the optimization sequence, a couple of test cases were configured and then the optimization sequence was performed for them. For this preliminary testing cases, Min-max was selected as the optimization algorithm. I/O interfaces between each pair of modules were implemented and checked carefully. In this way, the modules become stand-alone, with their coupling, for the purpose of performing a full calculation sequence, achieved via the I/O interfaces. Due to its modular feature, the developed optimization package can be directly employed in a parallel computational environment to reduce the search time. The particle transport module, that consumes the largest computational time, can be executed for a number of states concurrently on multiple processors. A distributed memory parallelism was applied to the code package via MPI (Message Passing Interface) [44] to execute this module for various states in the same search iteration on a number of processors concurrently. Furthermore, dynamic scheduling was enabled to enhance the load balance among the processors.

Another task was to test the code’s modularity. A consistency check was performed between the modules by replacing the contents of the modules. This way, the entire structure of the optimization package was tested and the I/O interfaces between each pair of modules were generalized. For this purpose, another optimization algorithm, Genetic Algorithms, was assembled to the code package. The performance of both optimization algorithms, in the code package were compared. Moreover, the optimization results of MOZAIK with two different optimization algorithms were obtained for the model problem, shape optimization of the D$_2$O moderator tank for the existing beam port configuration of PSBR. In this stage, some detailed calculations were also performed to improve the computational models for the model problem. In the particle transport module, the PSBR core was simulated by defining a neutron source at the core faces. Since the material distribution changes from one state to another as generated by the optimizer module, the effect of the material distribution near the core interfaces on the neutron angular, spatial, and energy distribution was investigated in order to define the neutron source accurately. In addition, a couple of new biased angular quadrature sets were developed and tested to improve the solution accuracy while reducing the computation time.
Finally, two new beam tube configurations were proposed for the PSBR and the code package was applied to determine the D₂O moderator tank shape for both configurations. After determining the layout of the new beam tube configuration based on purely geometric considerations, the characteristics of the neutron output of each beam tube were determined and related cost functions were defined. In order to compute the cost value of each state accurately, a streaming operator was implemented to compute the neutron beam intensity and spectrum at the exit end of each beam tube. The optimization calculation was performed for one beam tube at a time to optimize the moderator/reflector material size and shape. Finally, performance of each design was compared to the performance of the existing beam tube configuration.

The organization of this study is as follows. The description of the modular optimization code package is given in Chapter 2. Organization of each independent module in the optimization sequence, the interface blocks that couple the modules and provide modularity, parallel implementation of the code package, and detailed descriptions of the contents of each module are presented on this chapter. In Chapter 3, a comprehensive study performed to test and verify the physics module is given. The definition of the model problem and its computational models, the sub-modules or functions developed specifically for this model problem, their integration into the code package, some numerical results, and performance analysis of the computational models are presented in this chapter. Two preliminary optimization calculations are presented in Chapter 4. The definition of the optimization problems, preparation of the modules for the optimization sequence, and some performance results are presented in this chapter. In Chapter 5, optimization calculations performed by MOZAIK for the existing beam tube configuration of the PSBR and for the new beam tube designs are given in detail. Finally, Chapter 6 presents a summary of this study with some conclusions, and opportunities for further studies.
Chapter 2

Modular Optimization Code
Package: MOZAIK

2.1 Introduction

The Modular OptimizAtIon Code PacKage (MOZAIK) was developed in this work. It was basically designed as a modular optimization sequence, that performs state space search for shape optimization problems involving the radiation transport in the field of nuclear engineering. In this chapter, the code’s modular structure, its modules and their organization in the optimization sequence are described.

2.2 State-Space Search Representation of the Shape Optimization

Shape optimization deals with problems of the form: find an optimal shape in two or three dimensions according to certain requirements. Shape optimization problems are usually solved numerically by using iterative methods in which one starts with an initial guess for the shape, and then gradually evolves it, until it morphs into the desired optimal shape [3, 4].

One of the robust ways for solving the shape optimization problem is to define it as a state-space search[5]. State space search is a process widely used in the
field of artificial intelligence (AI) in which successive configurations or states of an instance are considered, with the goal of finding a goal state with a desired property.

An analogy can be drawn between the shape optimization problem and a state space search [10, 11]. In this analogy, the problem geometry is divided into cells and the material distribution in each cell is changed independently. In this way, a number of material distributions are generated and the best material configuration is sought that satisfies the given criteria.

In this analogy, a specific material distribution over the problem domain is called a state and it corresponds to a permissible shape of aimed component investigated. The collection of all states is called the search space. Usually, the search sequence starts from an initial state that is a guess, based on experience, or previous calculations. The evaluation or cost function is a function representing a weighted average of several criteria based on which the “goodness” of the shape (i.e., state) is quantified with the weight-reflecting the relative importance of each criterion. The objective of the search is to find a goal state that minimizes/maximizes the cost function over the entire search space. By this analogy, the shape optimization problem translates into a search for the goal state in the search space.

In this work, the term “optimal” is used in a loose way. Specifically, it indicated the material assignment to cells in the search problem domain that minimizes the cost function among all visited states during the search sequence. This is in contrast to the true optimal state that minimizes the cost function over the entire search space.

2.3 Introduction to MOZAIK

MOZAIK, basically designed as a modular optimization sequence, performs state space search for shape optimization problems involving radiation transport processes in nuclear applications.

In order to provide modularity and most general form of the code package, each MOZAIK module introduced in Section 2.5 was designed as an independent code package to perform assigned tasks sequentially. In addition, some fixed interface
blocks were defined among the modules to provide the modular operation of the code package.

A schematic of the basic modular structure defined for MOZAIK’s modules is shown in Fig. 2.1. In this configuration, each module, except the first in the execution sequence, has two interfaces which will provide its connection to the other modules:

1. **Input interface** is designed to provide required data to the module from the other modules.

2. **Output interface** is fixed to carry module outputs to the other modules.

![Figure 2.1. Modular Structure](image)

By this structure, a module becomes totally independent from the code sequence and the contents of the module can be easily switched as desired as long as the new module is designed to conform to these interface structures. The main disadvantage of this structure in the optimization sequence is that rebuilding (code compilation) of the whole code sequence is inevitable after modifying the contents of any module.

Originally, the modular configuration shown in Fig. 2.1 was employed the MOZAIK’s modules. Then, it was redesigned as shown in Fig. 2.2 to provide perfect modularity. In the new design, MOZAIK’s modules were perfectly separated from the optimization sequence by replacing input and output interface blocks...
with an input and output interface file, each possessing a specified fixed format. In other words, each module was redesigned as a separate or stand-alone code and each reads required data from the interface input file generated by the other modules and writes the module outputs to the interface output file for the other modules. A new module (executable file of module) can be easily replaced, without rebuilding MOZAIK’s executable, with the existing module only if it was designed as having an ability to read this input interface file and to write its outputs to the output interface using the required format. As a consequence of this structure, the code package serves the shape optimization calculations with a perfect modular feature and generic form.

**Figure 2.2. Module Structure in MOZAIK**

The organization of each module, their contents, and a description of the interface blocks among the modules are presented in Section 2.4.

### 2.4 Optimization Sequence

Initially, the code package was designed as having four independent modules: (1) initializer module, (2) physics module, (3) optimizer module, and (4) shape verifier module. A generic layout of the components of the initial optimization procedure is shown in Fig. 2.3.
In the original code sequence, the task assigned to the shape verifier module was to smoothen the produced optimal shape to make it more amenable to standard fabrication processes. In other words, it performs a post-processing operation on the results of the optimization procedure. Since the function of the verifier module is not integral with the optimization procedure, and because smoothing operators turned out to be far more complicated than originally perceived, the shape verifier module was removed from the code sequence. The so-truncated optimization procedure, including the other three modules as implemented in MOZAIK and exercised in this work is shown in greater detail in Fig. 2.4:

1. **Initializer module:** includes procedures for initializing the search process, and setting problem parameters for the other modules.

2. **Physics module:** performs the transport calculation for the model problem; in other applications not involving radiation transport this module would be replaced with a computer code modeling the corresponding physics.

3. **Optimizer module:** evaluates the physics calculation results via a given criterion and applies the optimization strategy to advance the search procedure to the next optimization stage.
Four fixed interface blocks, depicted in Fig. 2.4, were defined among the modules to enable the modular structure:

1. Interface-A carries problem parameters from the initializer modules to the physics and optimizer modules.

2. Interface-B was defined between the initializer and the physics modules to send the material distribution maps (arrays) of the initial states to the physics module.

3. Interface-C was defined between the optimizer and the physics modules to send the material distribution maps of new states to the physics module.

4. Interface-D was defined between the physics and the optimizer modules to
carry output quantities computed by the physics code to the optimizer module for evaluating the physics calculation results.

By using these fixed interface blocks among the modules, the code package attains its two most important characteristics: generic form and modularity. The benefit of this modular structure is that the contents of the modules can be switched depending on the requirements of accuracy, computational efficiency, or compatibility with other modules. In addition, this structure facilitates the parallelization of the whole optimization algorithm.

2.5 Organization of Modules

Each MOZAIK module was also prepared to possess a modular structure. In other words, each contains sub-modules which implement some special tasks assigned to it. By this feature, module contents become easier to understand, thus more amenable to future modifications.

2.5.1 Initializer Module

The initializer module gets data from MOZAIK’s main input processor and prepares data for the other modules in the optimization sequence. It consists of a couple of sub-modules as depicted in Fig. 2.5:

1. Input processor:
   Gets and processes data from MOZAIK’s main input processor.
   
   (a) Input Reading: Reads input data needed by the physics and optimizer modules.
   
   (b) Input Checking: Checks all entered input data. If one or more are inconsistent, it warns the user, suggests necessary corrections and terminates execution.
   
   (c) Input Echoing: Writes input data back to the output file in an appropriate format if the data is consistent.
2. Data-pool checker:
   All auxiliary files for the physics and optimizer modules such as state data, cross section data, mesh data, input templates, etc., reside in a specified directory in the file system called *data-pool*. This sub-module checks the necessary data for the physics and optimizer modules and verifies that it exists in the specified data-pool. If any data is missing from the data-pool,
the code warns the user of this deficiency and terminates execution.

In addition, the state data are stored in the data-pool during the optimization. Thus, MOZAIK can easily access these files if restart is requested by the user starting from the last successful iteration following an earlier unsuccessful iteration or unexpected job termination. Similarly, if the user provides the initial state data in the data-pool, the optimization sequence will be started from this data instead of an initial state in the optimization sequence.

3. Mesh data checker:
Checks the conformity of the fine and coarse computational mesh structures, if the dual mesh option, which will be described in detail in Section 2.7.3, is selected by the user. The dual-mesh module requires that both mesh data satisfy certain criteria to generate the transfer matrix of the material distribution map between the coarse and fine mesh structures. If the mesh data does not conform with the requirements, the module warns the user and terminates the job execution.

4. Initial state generator:
Originally, this sub-module was developed to generate the computational mesh for the given problem geometry and to prepare the material distribution array for the search procedure's initial state(s). In order to enhance modularity of the code package, this sub-module was split into two new functions because the initial state generation usually depends on the optimization algorithm:

(a) Generate the computational mesh for the given problem geometry and determine the size of the material distribution array for the x,y, and z dimensions. This sub-module does not generates computational mesh if the dual mesh option is enabled in MOZAIK's main input file. In this case, MOZAIK needs two different computational mesh structures provided by the user in the data-pool.

(b) Read the initial state data provided by the user from the data-pool. First the sub-module reads the size of the material distribution array
for the \( x \), \( y \) and \( z \) dimensions, then it compares these values with the values obtained from the computational mesh generated in the previous stage. If the values are equal, it allocates the material configuration array and assigns its material-index values by reading the state data from the initial state data file in the data-pool. Otherwise, it warns the user and terminates the job execution.

Although the new function enhances the modularity, it necessitates the preparation of the initial states separately from the code package. The initial state data should be generated using the optimization algorithm’s rules, and prepared using the appropriate file format.

Another important contribution of this function to MOZAIK is that it is compatible with the operation of restart module. For the restart case, the state data of a previous successful iteration is read by this module from the data-pool and the optimization sequence progresses into the next iteration.

5. Checking availability of hardware:
Most particle transport codes need large memory space to allocate the necessary data arrays and large disk space to perform I/O operations when memory space is insufficient to hold all data. Therefore, before starting the computation the code calculates the expected amount of requested disk space and memory usage, then compares them to the available resources on the target platform. If the computational node has insufficient resources, the code warns the user and terminates execution. Although this sub-module defeats the modularity of the initializer module because of its dependencies on the physics code, it is indispensable to start and continue the optimization without any problem related to the computational environment.

6. Interface file generator:
This sub-module generates the interface files using the appropriate format which will be discussed in the Section 2.5.4 for the physics and optimizer modules.
2.5.2 Physics Module

This module is executed repeatedly for each state evaluated in the optimization sequence. It consists of three major sub-modules as illustrated in Fig. 2.6:

![Diagram of Physics Module]

**Figure 2.6.** Contents of the physics module

1. Pre-processing:
   This module generates the necessary input file(s) for the transport code by using the state’s data provided by either the initializer module or the optimizer module. It creates a sub-directory and copies the created input file(s), binary particle-source file and binary cross section file to it. This way, each physics model run for each state will be performed in a separate directory thus avoiding interference between runs that may occur via common data or I/O files in the same directory.
2. Execution:
   This module includes only the transport code execution. The code is executed by using an appropriate system call. If execution is terminated unsuccessfully, this module sets a failure flag to this state and proceeds to evaluate another state in the optimization sequence. Otherwise, following successful completeness of a run the code proceeds to the post-processing stage.

3. Post-processing:
   This module processes the output files from the transport code in order to extract the desired values from the solutions that are subsequently used in computing the quantities to be optimized. By inserting user-specified input flags and data in the transport code input file, this sub-module selects appropriate data processing functions. Then, it reads all the data and prepares it in the requested format for compact output files and for the cost function calculation. After reading the solution from files, the sub-module removes all unnecessary files to save disk space.

4. Interface file generator:
   Generates the interface file for the optimizer module.

2.5.3 Optimizer Module

Different optimization algorithms can be inserted in this module. The user can choose the algorithm and can set its corresponding parameters in the main input file. The optimizer module consists of two main sub-modules:

1. Cost Calculation:
   This sub-module includes cost calculation in terms of the defined cost function and the selected optimization algorithm. The computed solution, e.g. group scalar flux in this study, for a set of states is transferred from the physics module to this module. Based on the user defined cost function and the state data, the cost value is computed for each visited state in a given optimization iteration. After sorting the states in terms of the magnitude of their cost, they are transferred to the cost evaluation routine.
2. Cost Evaluation:
Convergence of the optimization iterations is checked in this sub-module. The best cost computed for the current iteration is compared to the best cost value of the previous iteration. If the new best cost value, i.e. the best cost of the current iteration, is worse than the best cost value of the previous iteration, convergence is achieved. In this case, this sub-module records all states’ data into a file and sends the best state data with a stop signal to the main module. Otherwise, it sends sorted state data to the new state generation module.

3. New State Generation:
New states, i.e. material distribution maps, are generated in this module by using the rule(s)/operator(s) of the selected optimization strategy and data from the previous iterations’ best state. In some cases, this module can
produce discontinuous material distributions that are members of the search space but inadequate as optimal states, at least in the present application. Therefore, this module also evaluates the new geometry and material distribution to determine if these are acceptable or not. If the new configuration is not acceptable, this module chooses another one until it arrives at a geometry and material distribution continuous and acceptable. Moreover, each created state is compared to all previously evaluated states. If it is identical to a previously evaluated state, it is eliminated and a new states is generated. Otherwise, it is recorded into the state history file and stored in state arrays to be passed to the physics module. After generating all the new states to be evaluated, the code proceeds to the next iteration.

4. Interface file generator:
Generates the interface file for the physics module for the next iteration.

2.5.4 Interface Blocks among the Modules

Interface blocks or interface data files sustain the modular feature of the code package. These interface blocks establish a communication bridge among the independent modules. The details for these interface blocks are presented in this section.

1. Interface-A:
Communicator between the initializer and the physics/optimizer module. This block is generated by the initializer module to send the problem data to the other two modules. It includes the following parameters for the problem:

(a) Header section, includes some basic information such as date and time, username, working directory, name of the physics code and the optimizer, etc.

(b) Maximum number of optimization iterations, and maximum number of states to be evaluated (for array allocation),

(c) Mesh numbers for the coarse and fine mesh structures,

(d) Mesh data for both modules (both coarse and mesh data if dual mesh is selected),
(e) Location of the data-pool in the file system,
(f) Physics module data:
   i. name of the physics code and its auxiliary files such as input templates, cross section data file, source data file, etc.
   ii. names of the pre-processor and post-processor codes for the physics code if the physics module does not directly include this information,
   iii. options for the physics code,
(g) optimizer module data:
   i. names of the optimizer code and its auxiliary files,
   ii. options for the optimizer;
(h) Extra block reserved for future use.

2. Interface-B:
   Communicator between the initializer and the physics module. This interface is generated by the initializer module to send the initial state data to the physics module. For the shape optimization problem, the state data is a 3D array which includes the material numbers distributed over the problem domain by the algorithm used for initial state generation. The material numbers used for the state data must be consistent with the material numbers in the physics module and the optimizer module for the optimization calculation to proceed without any problem. Since the physics module reads the problem data such as mesh data, names of the codes, etc., from the interface-A block, this block only includes the material distribution of $M_0$ initial states. The first entry of the block is an integer flag that indicates the status of the data for the given iteration, 0 indicates this is initial data, and 1 indicates this is restart data. This information will be used by the multi-tasking module to manage the parallel execution of the physics module. In addition, the number of states, the size of the array for the material maps and its shape should be set in this block. A full listing of the contents of this block follows:
   (a) status (0 or 1)
(b) a simple header explains that this block includes state maps for iteration 
\( \text{ITER} \)
(c) optimization iteration number, \( \text{ITER} \) (0 for the initial states)
(d) number of states in the iteration \( \text{ITER}, \text{ISTATE} \)
(e) dimensions of the state map (material distribution) array, \( IM, JM, KM \)
(f) state number, 1
(g) state map \( (IM \times JM \times KM) \) space-separated integers
(h) state number, 2
(i) state map
(j) ...
(k) state number, \( \text{ISTATE} \)
(l) state map

During the optimization sequence, the number of states in each iteration can vary by depending on the optimization algorithm. Therefore, the size of this interface block or interface file might vary from one iteration to the other. For this reason, the number of states evaluated in the present iteration must be included by this file. The physics module reads all initial state data from this interface and generates its physics code inputs using them.

3. \textit{Interface-C}:

Communicator between the optimizer and the physics module. This interface is generated by the optimizer module to send the new state data to the physics module. Its structure is the same as the Interface-B except for the status flag. This flag is not necessary for this interface because it is only needed to initialize the parallel execution.

MOZAIK also uses the same structure to save the data for states examined in the previous iterations for providing the state data both for restarting the search procedure and for making comparisons with the new states generated by the optimizer module.
4. Interface-D:

Communicator between the physics and the optimizer modules. This interface block is generated by the physics module to send the solution computed by the physics calculation for all states to the optimizer module to perform the cost calculation and evaluation. The output quantity, i.e. the physics solution, might be a scalar quantity or a vector, i.e. one-dimensional array of scalars. Therefore, its array dimensions should be given in this interface block. The structure of this block is:

(a) a simple header explains that this block includes output quantities for the $ISTATE$ states in the iteration $ITER$,

(b) optimization iteration number, $ITER$

(c) number of states in block, $ISTATE$

(d) array dimensions of the output quantity $IDIM,I1,J1,K1,N1,M1$ (up to 5 dimensions, $IDIM=0$ if the quantity is a scalar)

(e) state number, 1

(f) quantity array

(g) state number, 2

(h) quantity array

(i) ...

(j) state number, $ITER$

(k) quantity array

MOZAIK also saves all output quantity arrays for an iteration for restarting purposes.

2.6 Parallel Implementation of MOZAIK

When implementing a parallel solution to a problem, one of the most important decisions is to determine the decomposition strategy, its suitability for the problem at hand, and its potential for delivering high parallel speedup at high parallel
efficiency. Then next, its components of the parallel code should be profiled and their contribution to good parallel performance is optimized. In other words, the problem is broken into discrete "chunks" of work that can be executed concurrently as multiple tasks on separate processors. This is known as decomposition or partitioning [45].

There are two basic ways to partition computational work among parallel tasks: domain decomposition and functional decomposition. In the domain decomposition technique, the data associated with a problem is decomposed. Each parallel task then works on the portion of the data assigned to it. In contrast, in the functional decomposition technique, the focus is on the computation that is to be performed rather than on the data manipulated by the computation. The problem is decomposed according to the work that must be done. Each task then performs a portion of the overall work.

Another important factor is the need for communication between tasks, and this usually depends upon the problem. Some types of problems can be decomposed and executed in parallel without need for tasks to share data. For example, Monte Carlo simulations for the solution of particle transport problems, wherein the same physical calculation is repeated for a number of particles denoted by \( N \). At the end of the physical simulations of its history particles’ contribution to a quantity, e.g. particle flux crossing a given surface, is calculated. So, the problem’s computational load can easily be distributed over \( M \) multiple tasks (processors) that then executed concurrently and independently of each other to do their portion of the work for \( N/M \) particles. At the conclusion of the simulation of all \( N \) particles, their contribution to the tallied quantity is collected from all \( M \) concurrent tasks. This type of problems is often called embarrassingly parallel because they are so straight-forward to parallelize and requires very little inter-task communication.

The next step is to determine the parallel programming model. There are several parallel programming models in common use: Shared Memory, Threads, Message Passing, Data Parallel and Hybrid [46]. Selecting a parallel programming model is dependent on the computational architecture and resources, and personal choice.

The Amdahl’s Law [45, 46]: The largest possible speedup is determines by the
fraction of code $P$ that can be parallelized:

$$speedup = \frac{1}{1 - P}$$

(2.1)

If none of the code sections can be parallelized, $P$ equals 0 and the $speedup$ equals to 1, in other words no speedup is achievable for this case. In the best possible case, $P$ equals 1 and the maximum $speedup$, is unbounded, i.e. approaches infinite, if the entire code is parallelized and executed over an unlimited number of processors.

Typically, however some sections of code cannot be parallelized due to their inter-task dependent structure. Therefore, the code can be divided into two parts: the parallel fraction of the code and the serial fraction of the code.

Introducing the number of processors, $M$, performing the parallel fraction of work, the relationship given in Eq.2.1 can be revised to:

$$speedup = \frac{1}{\frac{P}{M} + S}$$

(2.2)

where $S$ is the serial fraction, so that $S = 1 - P$. These fractions can be estimated by measuring the wall-clock time for various code sections in the serial execution of the code. It is obvious that there are limits to the scalability of parallelism whenever $S$ is not equal to zero.

The efficiency of the parallelism is calculated by the following formula:

$$Eff = \frac{speedup}{M}$$

(2.3)

2.6.1 Multi-tasking Module: Parallel Execution of MOZAIK

The state space search technique requires substantial computation time to accomplish the objective of the optimization by covering a broad range in the search space. Usually, the search is performed on a number of processors to reduce the total time of the optimization calculation.

MOZAIK is suitable for parallelization since the physics module computation of different states within the same iteration can be performed on a number of processors independently. Parallel execution of the physics module for different
states is expected to reduce the total computation time of the package by a factor of $M$, where $M$ is the number of parallel tasks (processors).

For this purpose, a multi-tasking module was developed and assembled to the code package to apply the distributed memory parallelism to MOZAIK via MPI [44]. A schematic of the serial and parallel versions of MOZAIK and the function of the multi-tasking module are given in Fig. 2.8. In MOZAIK, the parallel fraction of the code package is comprised of the physics module itself and the serial fraction is the rest of the modules as depicted in Fig. 2.8.

![Diagram of serial and parallel versions of MOZAIK](image)

**Figure 2.8.** Execution sequence for the serial and parallel versions of the MOZAIK.

Basically, this module includes some MPI functions to manage the processes in the parallel environment and to send/receive the data to/from the concurrent physics calculation. In addition, it has several data decomposition and distribution...
strategies which will be discussed in Section 2.6.2 designed to improve parallel performance.

The functions of this module are:

1. Master process:

   (a) Before starting the optimization, this function initializes parallel execution,

   (b) Receives the problem data from the initializer module and broadcasts the problem data to each processor in the parallel environment,

   (c) Receives the initial state data (data for $M_0$ states) from the initializer module, decomposes the state data and sends the individual data blocks (data for $PN_j$ states) to participating processors,

   (d) Receives the output data from each of the participating processors after they complete the physics calculations for the set of states assigned to each processor by the master process,

   (e) Checks the output data to verify that the physics calculation for each state was successfully completed. If the physics calculation for any state terminates unsuccessfully, the master process redistributes that state’s data to complete the physics calculation,

   (f) Combines the outputs and obtains an output array to feed into the optimizer module,

   (g) After getting the optimal result from the optimizer module, finalizes parallel execution.

2. Slave process:

   (a) Receives data from the master process,

   (b) Sends data back to the master process after slave finishes the physics calculation for states assigned to it by the master process.

In the optimization sequence, the code spends a significant amount time performing the physics calculation. By introducing the multi-tasking module to
MOZAIK, the total computation time consumed by the physics module is reduced by a factor close to \( M \), i.e. number of parallel tasks (processors). This also encourages the usage of MOZAIK for the shape optimization problem in nuclear applications because many codes related to radiation transport do not have parallel execution capability. MOZAIK’s parallel structure and the new organization of the modules for this structure are presented in the following sections.

### 2.6.2 Organization of Parallel Tasks: Data Decomposition Strategies

Amdahl’s law given in Eq.2.2 states that the speedup is almost proportional to the number of processors if ideal parallelism is achievable. However, in fact it is not realistic to obtain speedup exactly proportional to the number of processors because of the existence of the serial sections of the code and the parallelism overhead, including communication cost, or penalty, during parallel execution.

In order to improve the parallel performance of MOZAIK we have to focus on its parallel section. The speedup of MOZAIK’s parallel execution increases, if the wall-clock time of the parallel physics calculation is reduced. Therefore, several data decomposition and process management strategies were developed and tested within the code package to minimize the overhead of parallelism and maximize parallel efficiency.

In MOZAIK, the multi-tasking module distributes the \( N \) state data to \( M \) processors for performing their physics calculations. For different cases having different number of processors, the data distribution to the processors affects the parallel performance of the code. A study was done to understand these effects. Two possible situations, demonstrated in Figs. 2.9 and 2.10, were considered: (1) the physics calculation of all states poses an identical computational load, and (2) the physics calculation of all states requires different execution time, e.g. different number of iterations till convergence of the transport code’s inner iteration.

1. The case in which each state’s physics calculation poses an identical computational load:
The example given in Fig. 2.9 illustrates a simple search-state decomposition across various participating processors. In the example, there are seven states, and supposing the computation time of each state’s physics calculation is one unit. For the sequential calculation, the physics calculation is completed sequentially for each state and the wall clock time for this calculation is 7 units. The reference time is taken as the total computation time of the sequential calculation, which equals the sequential wall-clock time of 7 units.

(a) Case-1: The number of processors equals the number of states (N=M):

This case is an ideal case. The execution time for each processor is same, and the wall-clock time equals the CPU time of a state/processor. For this case, the speedup and the efficiency are (supposing the serial fraction of the code and the communication penalty are negligible):

$$speedup = \frac{t_{sequential}}{t_{parallel}} = \frac{7 \ units}{1 \ unit} = 7$$ \hspace{1cm} (2.4)

where $t_{sequential}$ is the wall clock time of the serial execution, and $t_{parallel}$ is the wall-clock time of the parallel execution for the same problem.

$$Eff = \frac{\text{speedup}}{N} = \frac{7}{7} = 1.$$ \hspace{1cm} (2.5)

(b) Case-2: The number of processors is greater than the number of states (M>N):

In this case, $N$ state data is distributed to the first $N$ processors. $N-M$ processors become idle during the calculation. This adversely affects parallel performance of. For this case, the speedup, the efficiency and the idleness are (supposing the serial fraction of the code and the communication penalty are negligible):

$$speedup = \frac{t_{sequential}}{t_{parallel}} = \frac{7 \ units}{1 \ unit} = 7$$ \hspace{1cm} (2.6)
Figure 2.9. Strategies for distributing tasks in the same optimization iteration; states are identical

\[ Eff = \frac{\text{speedup}}{N} = \frac{7}{10} = 0.7 \quad (2.7) \]

\[ Idleness = \frac{t_{idle}}{t_{parallel}} = \frac{3/10}{1} = 0.3 \quad (2.8) \]

where \( t_{idle} \) is the average idle time for the processors during the calculation.
(c) Case-3: The number of processors is smaller than the number of states (M < N):

In this case, the speedup equals to M and the efficiency is 100% if \( N \) exactly divides \( M \). Otherwise, the first \( N/M \) states data is distributed to each processors. Then, the remaining data is distributed to some of the processors. Although each state calculation consumes the same amount of time in this case, individual processor’s loads become unequal due to the uneven work distribution. Because, namely after completing their assigned calculations, processors 1 and 2 have to wait for processor 0 to finish its assigned work. For this case, the speedup, the efficiency and the idleness are (supposing the serial fraction of the code and the communication penalty are negligible):

\[
\text{speedup} = \frac{T_{\text{sequential}}}{T_{\text{parallel}}} = \frac{7 \text{ unit}}{3 \text{ unit}} = 2.33
\]  

\[
Eff = \frac{\text{speedup}}{N} = \frac{2.33}{3} = 0.778
\]  

\[
\text{Idleness} = \frac{t_{\text{idle}}}{t_{\text{parallel}}} = \frac{2/3.}{3.} = 0.222
\]

2. The physics calculations for different states pose different computational loads:

Optimization calculations with MOZAIK are similar to this case since the computational time of each state, depending on the material distribution (state data), is different. Moreover, the case (M > N) is not a reasonable setting for MOZAIK calculations given in this study because of the typical limited computer resources and the large number of states evaluated per MOZAIK iteration. Hence, only the cases (M = N) and (M < N) are considered for improving the parallel performance of the code package for this type of situation.

The example given in Fig. 2.10 illustrates some simple states decomposition attempted for this class of cases. In the example, there are seven different states, each requiring a different computation time by the physics model ranging from 0.5 to 1.5 time units.
Figure 2.10. Strategies for distributing tasks in the same optimization iteration; different states pose different computational loads

(a) Case-1: The number of processors equals to the number of states (N=M):

The results of this case show that the parallel performance of this case is totally different from the performance of the same case in the previous example. Six processors wait after finishing their jobs until processor
“0” finishes its computation. Thus the idleness is significant in this case and it directly affects parallel performance.

(b) Case-2: The number of processors is smaller than the number of states (\(M < N\)):

In this case, four different scheduling approaches were tested for distributing the states among the participating processors with the objective of minimizing the idleness:

i. Static Scheduling:
Assigns tasks before execution commences. This algorithm distributes the first \(N/M\) states among the participating processors before executing the physics code in parallel. Then, it distributes the remaining states to some of the processors. The computational load for each processor varies depending on the number of physics calculations, i.e. states, assigned to it, and the computation time consumed by each state. Although the results show that this kind of scheduling is not the best way to adjust the load balance among the processors, it might be improved by increasing the number of states in an optimization iteration.

ii. Dynamic Scheduling-I:
Assigns tasks during execution. In this algorithm, one state is assigned to each processor by the master process. After all processors, slaves and master, finish their first set of assigned physics calculations, a second set of state data is assigned to them by the master process. This continues until all states are distributed and computed. In this scheme while at least one processor is still running, the remaining processors have to wait until all active physics calculations are completed in each states distribution cycle. Idleness is evident in each states distribution cycle of this algorithm.

iii. Dynamic Scheduling-II:
Assigns tasks during execution. In order to improve the parallel performance of the previous algorithm, one of the processors, executing the master process, is dedicated to manage the parallel tasks
during the progress of the calculation. This way, a new state is immediately assigned to a slave processor by the master process once the slave process finishes its physics calculation for the previous state. This strategy shifts the idleness of the processors to the end of the calculations for an iteration. Nevertheless, the parallel performance is not good for this algorithm due to the idle time incurred by the master processor as it manages the parallel execution of the slave processes. This might be useful if the calculation is done with many slave processes since the master processor’s idle time as a fraction of the total execution time decreases.

iv. Self Scheduling:
A form of dynamic scheduling where the processors themselves select which state to execute next. In this algorithm, the entire set of states within an iteration is sent to all processors by the master processor, and then each processor (both master and the slaves) selects its own state from the entire set. After making the selection, the processor marks the selected state to exclude it from being selected by any of the other processors. Execution for this iteration will be finished after the calculation of all states is completed. This strategy reduces the idleness significantly during the calculation as shown in Fig. 2.10.

This study reveals that good parallel performance for MOZAIK can be obtained by assigning $M$ states to $M$ processors whenever the physics calculation of all states consumes the same amount of time. As a matter of fact, this constraint is not realistic for the optimization calculations with MOZAIK that will be presented in Chapters 4 and 5. Therefore, the self scheduling and static scheduling algorithms were implemented in the multi-tasking submodule not only to use the available computational resources fully, but also to improve the parallel performance of MOZAIK.
2.6.3 Modules Organization for Multi-processor Environment

Originally, four scheduling algorithms were implemented in MOZAIK to improve its parallel performance during the optimization calculations. Subsequently, two of them, dynamic scheduling I and II, were removed due to their poor performance and only the static-scheduling and self-scheduling algorithms were retained in MOZAIK. As a result, by using these scheduling algorithms the master processor is assigned for not only managing parallel execution but also running the physics module for a set of states.

1. MOZAIK with the static scheduling algorithm:

   This is MOZAIK’s default scheduling strategy if it is executed in a parallel environment. A schematic of the new organization of MOZAIK’s modules for this parallelization is depicted in Fig. 2.11. Initially the section of the multi-tasking module, which was designed for the master process, reads the interface blocks from the initializer module and broadcasts the problem data to all slave processes. Then it arranges $M$ data sets from all states’ data, sends $M - 1$ slave processes and retains one set for itself. Next, the master process, as well as the slave processes proceed to conduct the physics calculations each on its own data set.

   After finishing its jobs, the master process opens receiving interfaces from the slave process to collect the outputs of each slave’s assigned states. It merges $M$ output arrays into one array and prepares an interface file for the optimizer module. It repeats the same procedure with the new data from the optimizer module until the optimizer module determines an optimal state has been achieved and terminates the execution of the code package.

   The code sections work of a slave process only reads a set of states data, prepares the interface file for the physics module, executes the physics module, reads the states’ outputs from the interface file generated by the physics module, and sends back the states’ outputs to the master process.

2. MOZAIK with the self scheduling algorithm:

   Applying the self-scheduling algorithm to the distributed memory parallelism
needs to be much more careful since in this algorithm each process selects its task from a state pool and repeats this action until all states in the pool are consumed. First of all, in an optimization iteration, a shared task pool or states pool should be generated and the accessibility of all processes to this data pool should be provided. This way a process either removes the state from the pool or marks the data in the pool as processed, after it selects it. A shared data lock file is also necessary to lock access to the data pool during the state selection process.

A similar strategy was used to implement the self-scheduling to the multi-
tasking sub-module. A schematic of the new organization of MOZAIK's modules for this parallelization scheme is depicted in Fig. 2.12. The master process section of the multi-tasking module generates the state pool which includes only state numbers in the ongoing iteration, after getting the state data either from the initializer module as the initial states or from the optimizer module as the new states. Then, it broadcasts the problem data and all states data to each participating processor. Finally, it invites all slave processors to start selecting the states from the state pool.

When a processor (either master or slave) is available to select a new state:

(a) First, it checks the status of the state pool access.
(b) Locks the state pool by accessing the data lock file, if it is available. Otherwise, it checks its status regularly until it becomes available.
(c) Selects the topmost unprocessed state and marks it as processed.
(d) Unlocks the state pool by changing the status of the data lock file.
(e) Generates an interface file for the physics module, and executes the physics module.
(f) Appends the physics output of this state to an array which will be sent to master process after all states are processed.
(g) Repeats the process from item a until all states in the state pool are marked processed.

After the physics calculation of all states are completed, the master process gathers the data from each slave process and generates an interface file for the optimizer module. This procedure is repeated until the optimizer determines an optimal shape.

The parallel performance of MOZAIK with the two scheduling strategies described above measured while performing the optimization calculation for the selected model problem will be given in Chapter 4.
2.7 Additional Features of MOZAIK

Although MOZAIK’s three modules were designed as having independent structure, it also has some independent modules, basically utility functions, which were embedded in the code package. It is not necessary to switch the contents of these modules when changing the contents of the three primary modules described above.

The functions of these modules are:

1. **Restart module**: Restarts the optimization from the last successful iteration.

2. **Multi-tasking module**: Enables multi-tasking support for MOZAIK.
3. **Debugging module:** Echoes additional messages for debugging purposes while code is running.

4. **Dual mesh module:** Enables employing two different mesh structures for the physics calculation and the other for the optimizer.

In this section the detailed description of these modules is presented.

### 2.7.1 Restart Module: Option “restart”

During an optimization calculation, unexpected job termination can result in substantial loss of intermediate data and the time and computational resources consumed until the moment of failure. Such catastrophic failure due to some technical problem in the computer system, e.g. the power outage, failure of one or more computational nodes, etc. Therefore, “restart” capability was added to MOZAIK by implementing the restart module. This module allows MOZAIK to restart from the last successful iteration if all the state data of this iteration is saved by the user in the MOZAIK’s data-pool.

The restart module checks the data of all states for the last iteration and determines the state whose calculation of which state has failed. Then, it copies the data of the uncompleted states to the working data-pool for the initializer module. The initializer module proceeds with employing the calculation with these state data. If the restart module does not find the required data in the data-pool for restarting, the code warns the user and terminates execution.

### 2.7.2 Debugging Module: Option “debug”

One of the most important features of a large code system is its debugging tools. Sometimes it is needed to follow the optimization calculations in detail or to understand the situation if an unexpected problem occurs such as code crash. For this purpose, a debug module was developed and installed in MOZAIK. This feature is activated by entering an event number (a set of events were considered and different number was assigned for each event such as event 1011: write initial state data to the output in ASCII) in the main input. Then, MOZAIK echoes some details for the requested event such as the contents of each interface block.
for each iteration printed in ASCII, ASCII plot of the new states generated by the optimizer, etc. The debug module can be easily modified and integrated with the three primary modules.

2.7.2.1 Simulate Physics Option

This option is part of the debug module; it enables execution of MOZAIK without any physics calculation. It was developed and installed in MOZAIK for reducing the computation time of the optimization sequence, down to almost 0.1 sec, for testing the new optimizer module in the sequence. In this way, a number of optimization iterations can be completed in a couple of seconds and the optimizer can be easily examined.

2.7.3 Dual Mesh Module: Option “Dual Mesh Support”

In MOZAIK, shape optimization is defined as a state space search by dividing the problem geometry into cells. Therefore, the size of the search space depends on the cell numbers in this translation. The major disadvantage of the larger search space is the longer computation time for the optimization calculations.

In some cases, the physics code needs a very fine mesh structure either to compute the output quantity more accurately or to define the geometry realistically. In the optimization sequence, the finer mesh structure yields an extremely large number of states to be evaluated, and as a result consumes excessively long computational time.

In order to reduce the computational time of the optimization calculations while executing the physics code with the fine mesh structure, the dual mesh module was developed and installed in MOZAIK. This module regenerates the interface file D between the optimizer and the physics modules. For this sub-module’s operation, two mesh files should be provided by the user:

1. Coarse mesh file for the optimizer module.

2. Fine mesh file for the physics module, that is generated by refining the coarse mesh structure provided to the optimizer module.
The dual-mesh module maps the state data from the optimizer module (state data for coarse mesh structure) to the state data for the physics module (state data for fine mesh structure). If only one mesh file is provided, MOZAIK uses the same mesh file for the physics and optimizer modules. Furthermore, the code warns the user and terminates execution if the two provided mesh files are inconsistent.

2.8 Physics Code: TORT

Oak Ridge National Laboratory’s TORT [14] code was selected as the transport solver in MOZAIK. TORT was especially designed for applications involving the deep penetration of targets by neutrons and photons. TORT can also be used for reactor eigenvalue problems.

In TORT, the steady-state linear Boltzmann equation is solved using the method of discrete ordinates to treat the directional variable and weighted difference, linear-nodal, or linear-characteristic method to discretize the spatial variables [12, 14]. A multi-group formulation is used to treat the energy dependence. Starting in one corner of the mesh, at the group with the highest energy, and with a starting guess for implicit sources, the specified boundary conditions and recursion relationships resulting from the numerical methods’ formalism are used to sweep into the mesh along each discrete direction[14]. Integral quantities such as the scalar flux are calculated as weighted sums of the directional results over all discrete angles in the employed angular quadrature. The calculation then proceeds to lower energy groups.

Iterations are used to resolve implicitness of the solution caused by scattering between directions within a single energy group, by scattering from one energy group to another previously calculated group (i.e. up-scattering), by fission, and by certain boundary conditions. Methods are available in TORT to accelerate iterative convergence. Fixed sources can be specified at either external or internal mesh boundaries, or distributed over the volume of the mesh cells. Either cylindrical ($R\theta Z$) or Cartesian ($XYZ$) geometry is supported, as well as several two-dimensional geometries. The choice of numerical method from among the three available options in TORT depends on the computational cost, numerical performance and the type of problem being solved.
By selecting the proper solution method, a suitable mesh structure and angular quadrature, TORT can compute the desired solution with sufficient accuracy in reasonably short execution times.

2.8.1 Mesh Generation

In the deterministic transport solver, the computation time is approximately proportional to the number of computational cells. If the number of cells is reduced, the computation time might be reduced whereas the iterative convergence rate decreases because of the increasing optical thickness of the cells. Therefore some additional outer/inner iterations might be necessary to avoid false convergence, thereby causing the computational time to increase.

The accuracy of the solution is also dependent on the mesh size. In a coarse mesh calculation, large flux gradients can adversely affect solution accuracy. Typically solution accuracy improves with decreasing mesh size once the numerical scheme enters the asymptotic regime with respect to mesh refinement level. Therefore, an optimal mesh size should be selected to balance the desire to decrease the computational time while achieving this highest possible solution accuracy. In addition, by refining the mesh, slanted and curved surfaces that do not conform to the employed Cartesian mesh for the problem geometry can be modeled in better detail and with higher fidelity.

For this reason, mesh size selection plays an important role in achieving iterative convergence and high computational efficiency.

2.8.1.1 GGTM Code

GGTM is a member code of the BOT3P code package developed at ENEA-Bologna Nuclear Data Centre to give the users of the DORT and TORT deterministic transport codes some useful diagnostic tools to prepare and check their input data files [47].

GGTM is a mesh generation tool for TORT. It generates the computational mesh using input data provided by user. In the input data, the coordinate system can be specified either X-Y-Z or R-Θ-Z. The geometry can be defined by using a combinations of windows each defining a geometric shape or body, e.g. rectangular...
co-ordinate window, cylindrical window, or ellipsoidal window. In addition, the mesh grid can be refined in each dimension separately within the specified regions. GGTM is also capable of generating a distributed source for the given geometry.

2.8.2 Cross Section Library Preparation: GIP

Solving the particle transport equation also requires providing the necessarily cross section data to the transport solver in an appropriate format.

GIP, the Group-organized cross section Input Program, was developed and subsequently included in the to DOORS package in order to provide cross section data to DORT and TORT [48]. GIP accepts nuclide-organized microscopic cross section data either from the input stream in card-image or from a data library prepared by the ALC1 program. GIP also accepts the cross sections from any ANISN formatted library. GIP allows the user to mixes the cross section data into macroscopic cross sections and to store them in a binary file. Furthermore, it can prepare cross section libraries for either forward or adjoint execution of the transport codes.

2.8.3 Quadrature Selection

In order to obtain accurate solutions by the discrete ordinates transport solver, the selection of the angular quadrature plays an important role. The usage of higher order angular quadratures increases the solution’s detail at small surface area located far from the source at which the flux is calculated and reduces undesirable ray-effects in weakly scattering configurations. On the other hand, it’s usage increases the computational time since the computational time of TORT grows with the number of angles in the angular quadrature set employed.

For this reason, the angular quadrature should be selected carefully to obtain more accurate results in reasonably short execution times.

There are several techniques for the generation of discrete ordinates and their associated weights [13, 12]. Level symmetric quadratures (LSQ) are most widely used for general applications. Another preferred quadrature set is the Square Legendre-Chebyshev (SLC) Quadrature set. The $\eta$ levels are set on the z-axis equal to the roots of Legendre polynomials and the azimuthal angles on each level
are calculated as the roots of the Chebyshev polynomials.

2.9 Optimizers

In order to test the modularity and finalize the codes modular structure, two different optimization algorithms were implemented in MOZAIK’s optimizer module.

2.9.1 Cost Function

The cost or objective function is a function associated with an optimization problem which determines how good a solution is. The cost function can be easily defined if only one parameter is modified to lower the cost or increase the benefit. Otherwise, most optimization problems are more complicated. Some optimization problems can involve thousands of variables requiring sophisticated mathematical analysis to define the cost function and achieve a successful result.

In MOZAIK, the cost function is defined as a part of the optimizer module. It is primarily dependent upon the state output data computed by the physics module and subsequently on the shape of a given state in the optimization sequence. The optimization algorithm is just used to generate new states using its operators. Therefore user of MOZAIK can integrate the cost function as a separate submodule without changing the contents of the optimizer.

2.9.2 Min-max with Depth-first Search Algorithm

For the shape optimization problem, the Min-max with depth-first search algorithm was selected as an initial optimization algorithm. The Min-Max algorithm is usually applied in two-player games, such as tic-tac-toe, checkers, chess, go, among others.

In Min-max, after generating the search tree by depth-first search algorithm, the value of each node in the tree, i.e. state, is calculated as illustrated by the integers in the example shown in Fig.2.13.a. Then, a search path is determined from the root node to the goal node as seeking the maximum value for each position in the same path. The disadvantage of this algorithm is that the computation cost increases proportionally with the size of the search tree. It is not realistic to
generate all branches and all nodes in each branch in the search tree and evaluate their values. One practical alternative approach involves pruning as depicted in Fig. 2.13.b:

1. First all possible nodes are generated along each possible branch (level 1).
2. The cost values of all generated nodes are computed.
3. The node with highest cost function value among the generated nodes on the current level is selected.
4. All the other nodes are discontinued, or "pruned”.

5. New nodes are generated from the selected node to compose a new level (level 2).

6. This process is repeated until the goal state is reached; the goal state is a node from which all generated nodes in a new level have lower cost values than its own.

Similar approaches were implemented in MOZAIK’s optimizer module. Two search operators were developed to generate new states: (1) with pruning, and (2) without pruning.

Min-max without pruning starts the calculation from an initial state at which the entire search problem domain, i.e. the set of computational cells whose material composition is manipulated to optimize the shape, is filled with only one material denoted A. To generate new states during the search, the material content of one of the A cells is replaced with material B cell. During the optimization, Min-max takes the best state of the previous iteration and performs the replacement operation on either a single A cell or a group of A cells around the B cluster (A cells are replaced with B) by using a breadth-first search strategy to obtain new states. Finally, Min-max yields a B and A material assignment distribution over the cells in the search domain, i.e. an optimal geometric shape, which maximizes the output quantity. This approach is illustrated in Fig. 2.14.

In contrast, Min-max with pruning, that is depicted in Fig.2.15, marks all newly added A cell in the search domain in a given iteration. Then in the next iteration it starts with the state with this marked cell, and performs replacement operation on either a single A cell or a group of A cells around the marked cell by using a breadth-first search strategy to obtain the new states comprising the new iteration. In this way, the new states only develop around the best state determined in the previous iteration. Therefore, in this algorithm the number of states in each iteration is smaller than that of in Min-max algorithm without pruning.
Figure 2.14. Replacement Operation in Min-max (without pruning)
<table>
<thead>
<tr>
<th>Initial State</th>
<th>Neighbours (Min-max-II)</th>
<th>Best State</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 0</td>
<td>6 New States</td>
<td>State 5</td>
</tr>
<tr>
<td>State 0</td>
<td>4 New States</td>
<td>State 3</td>
</tr>
<tr>
<td>State 0</td>
<td>4 New States</td>
<td>State 1</td>
</tr>
<tr>
<td>State 0</td>
<td>5 New States</td>
<td>State 1</td>
</tr>
</tbody>
</table>

Figure 2.15. Replacement Operation in Min-max (with pruning)
2.9.3 Genetic Algorithms

A genetic algorithm (GA) is a search technique used to find exact or approximate solutions to optimization and search problems. Genetic algorithms are implemented as a computer simulation in which a population of abstract representations, called chromosomes, of candidate solutions, called individuals (in this study called material distribution maps or states), to an optimization problem evolves toward better solutions. The evolution usually starts from a population of randomly generated individuals. In each generation, the cost of every individual in the population is evaluated, then multiple individuals are stochastically selected from the current population (based on their cost), and modified (recombined and possibly randomly mutated) to form a new population.

In contrast to the Min-max replacement operator, the genetic algorithm has three operators to generate new states. Four basic genetic operators, each has several functions, were defined for MOZAIK’s optimizer module:

1. Initial Population Generation:
   (a) Random distribution: All material cells within the problem domain are randomly assigned materials A or B.
   (b) Random Walk: A 3D pattern is created by performing random walk in the problem domain, then material A is placed in this pattern.
   (c) Depth-first tree: A 3D pattern is created by using Min-max’s replacement operator developed for MOZAIK for several iteration, then material A is placed in this pattern.

2. Selection Operator:
   Selects two individuals for the mutation and cross-over operators by one of the following rules:
   (a) Select two individuals randomly.
   (b) Take the best individual (state) of the previous iteration and selects one individual randomly.
   (c) Group the individuals according to their cost values. Then randomly select two groups, and select their best individuals.
(d) Group the individuals according to their cost values. Then randomly select one group, and selects their best individual, take the best individual (state) of the previous iteration as the second individual.

3. Cross-over Operator:
Marks a region, then swaps the contents of this region between two individuals. Uses the following rules for marking the region:

(a) One-point: randomly select a point in the 3D grid denoted by its indices \(i,j,k\). Then mark a region from 1 to \(i\), 1 to \(j\), and 1 to \(k\).

(b) Two-point: randomly selects two point in the 3D grid denoted by their indices \((i_1,j_1,k_1)\), and \((i_2,j_2,k_2)\). Mark the region from \(i_1\) to \(i_2\), \(j_1\) to \(j_2\) and \(k_1\) to \(k_2\).

(c) Random Walk: Mark a region by performing a random walk in the 3D grid (both self-avoiding and non self-avoiding)

(d) Depth-first tree: A 3D pattern is created by using Min-max’s replacement operator developed for MOZAIK for several iterations. Subsequently, this pattern is marked as a swapping region.

4. Mutation Operator:
Randomly selects one cell in the 3D grid, and invert its material content from one material to the other. Similarly, it can perform the same strategy on a group of cells, clusters.

GA designed for the optimizer module of MOZAIK starts the optimization calculations with an initial population (a set of initial states). After getting the output quantity for each state that was computed by the physics code, it calculates the cost of each state using the defined cost function. Its selection operator selects two individual (two states) according to their cost values with the given requested strategy. Then, its crossover operator performs its selected swapping strategy to generate two new individuals. In GA developed for MOZAIK, each crossover operation is followed by a mutation operation with a mutation probability which was designates as 0.7. After generating new individuals (states), the data send to physics module for obtaining necessary output quantities for the cost function. This procedure is repeated until the goal state is reached.
Model Problem Development: Verification Studies for the Physics Module

A set of computational models based on the existing beam port configuration of the Penn State Breazeale Reactor (PSBR) was designed to test and validate the selected physics code, TORT, and the requisite data such as source distribution, cross-sections and angular quadratures. These models were also used to develop some tools essential to the physics module for the PSBR’s design calculations. In addition, with these computational models the functionality of the physics module was validated in the optimization sequence. These studies illustrate that the physics module with TORT is suitable for inclusion in MOZAIK to conduct further optimization calculations for the PSBR’s beam port facility.

This chapter presents the neutronic models developed, the TORT validation calculations, and performance evaluation for these models. In addition it describes some tools originally developed for these calculations and elaborate their utility in assisting the physics module during the optimization calculations.
3.1 An Overview of PSBR

The Radiation Science and Engineering Center (RSEC) facilities at The Pennsylvania State University include the Penn State Breazeale Reactor (PSBR), gamma irradiation facilities, and various radiation detection and measurement laboratories. One of the experimental facilities at the RSEC is the Neutron Beam Laboratory (NBL) including seven beam ports, two of which are currently used for neutron transmission and neutron radiography experiments.

The PSBR core is itself a TRIGA Mark-III research reactor design licensed for the steady-state operation at powers up to one mega-watt (MW) and with a pulsing capability to 2000 MW [25]. The reactor core operates in a pool of demineralized water at a depth of about 5.5 meters that provide the necessary shielding, cooling, and reflection for reactor operation.

The PSBR core is moveable: it can be translated in the east-west and north-south directions, and it can be rotated in the azimuthal direction providing much flexibility in the positioning of the core for experiments. The core is coupled to a D$_2$O tank and graphite reflector block to provide thermal neutron beams to the existing beam ports, BP-4 and BP-7, respectively. A schematic for the existing beam port arrangement of PSBR within the biological shield is given in Fig. 3.1.

The vertical locations of the seven beam ports vary relative to the centerline of the TRIGA reactor’s fuel rod’s height. Only BP-4 is located at the centerline of the TRIGA core. Beam ports BP-1, BP-3, BP-5 and BP-7 are 12.7 cm below the core centerline, and BP-2 and BP-6 are 27.9 cm below the core centerline as depicted in Fig. 3.2. Due to this beam holes’ configuration and the design of the current moderator tank, only BP-4 and BP-7 are functional for experimental purposes.

A set of neutronic models presented in the next section were designed based on the configuration of the existing primary beam port BP-4, to test and verify the transport code TORT, the multi-group cross section library, and the entire MOZAIK’s physics module.
Figure 3.1. Top View of the Existing PSBR Beam Port Layout

Figure 3.2. PSBR beam port locations relative to TRIGA fuel rod’s height
3.2 Computational Models

Originally, Sarikaya and Alim developed an MCNP model including the PSBR core, the D$_2$O moderator tank adjacent to the core, the graphite reflector block behind the moderator tank, and the other components of BP-4 and BP-7 [28, 29, 30]. A schematic of this model is depicted in Fig. 3.3.

This MCNP model has two major disadvantages: (1) The neutron and gamma outputs at the beam tube exit end (i.e. the beam tube exit hole) have very large variance due to the poor statistics, and (2) Extremely long execution times are necessary to tally any neutrons at the beam tube exit end. Therefore, Sarikaya and Alim divided this model into two parts and performed their calculations separately with the two new models [28, 29, 30].

Figure 3.3. Schematic of the full model of the PSBR and beam tubes developed by Sarikaya and Alim
As a consequence of their effort, it became obvious that model-splitting is essential to achieve accurate results in a reasonable amount of time. In this study, the model problem was divided into three pieces. In this way, only one of these neutronic model pieces was provided to MOZAIK to perform the optimization calculations for the PSBR beam port configuration.

The schematics of these three models, Core Model, Tank Model, and Beam Tube Model, are shown in Fig. 3.4. The following three sections briefly describe these computational models and their roles in the PSBR’s optimization calculations.

![Figure 3.4. Computational models obtained by separating the full neutronics model for BP-4 into three pieces](image)

### 3.2.1 Core Model: MCNP Model of the PSBR and BP-4

*Core Model* was developed only for the MCNP5 code. Mainly, this model comprises a large water pool which contains the following components:

1. PSBR core detailed model provided by Tippayakul [23, 28, 29].
2. A drum shaped D$_2$O moderator tank adjacent to the PSBR core.
3. A graphite block behind the D$_2$O moderator tank.
4. Beam re-entry gap in the D$_2$O moderator tank.

5. Bismuth gamma shielding disk placed at the junction between the beam tube and the beam re-entry gap.

MCNP core calculations were performed with this model to compute the neutron flow across interfaces at specified locations/surfaces in full spatial, angular, and energy detail. The major task of this model was to provide the neutron data at the PSBR core-D$_2$O tank interface to generate a boundary neutron source for the Tank Model. In this way, the Tank Model enabled the efficient computation of the neutron output after the Bi disk without performing repetitive core calculations. This was designed to reduce the computational time significantly. This MCNP model also provides the neutron outflow after the bismuth gamma shielding disk that could be used in the validation exercise of the Tank Model as described later.

In addition, this model was used for the verification studies performed once optimization calculations was finished. In the optimization problem presented in Chapter 5, a moderator tank shape was investigated to maximize the thermal neutron output at the beam tube exit. Therefore, by assembling the optimal moderator tank configuration determined by MOZAIK for this problem, a set of results were obtained by MCNP calculations to verify the optimality of the results and to evaluate the optimal tank shape’s neutronic performance.

For this reason, the MCNP Core Model was designed to possess a fixed part and a modular part:

1. Fixed part:
   The fixed part of the model includes only the PSBR core.

2. Modular part:
   The other components such as moderator tank, graphite block and the beam port components were designed as modular components. In this way, one of the components can be easily removed from the model or moved from its original location to another location. This modular part is automatically generated by the code gentinp-3 originally developed as part of a pre-processing sub-module of the physics module to generate the TORT input file. Subsequently, the MCNP input generation capability was added to this sub-module
to obtain both TORT and MCNP input files for the same geometry and material distributions.

The important advantage of gentinp-3 code is that exactly the same geometry is utilized to generate the corresponding input for both codes thus improving consistency between the executed models. Sample geometric configurations suitable for input into each of the two transport codes as generated by gentinp-3 during the optimization calculations is shown in Fig. 3.5.

*Figure 3.5.* Model geometries generated by gentinp-3 for TORT and MCNP codes (Model at the left side is MCNP5 Core Model, model at the right side is TORT Tank Model)
This input generator, gentinp-3, has the following features:

(a) Compatible with the mesh generating tool for TORT, GGTM3 [47].

(b) Includes a set of level-symmetric and biased angular quadrature sets and also accepts user provided angular quadrature sets to generate TORT input file.

(c) Reads TORT mesh structure and TORT input template, generates TORT input file.

(d) Reads TORT mesh structure and MCNP input template, generates MCNP input file.

(e) Defines mesh-tally for any surface for the MCNP models.

(f) Makes co-ordinate transformation from one model to another model.

(g) Compatible with MCNP’s coordinate transformation feature, lattice declarations, and macro-body declarations [17].

The results of the MCNP Core Model calculations are presented in Section 3.7. The MCNP Core Model also includes MCNP5’s SSW card (Surface Source Write [17]) which is used to obtain neutron histories data at the specified locations to compute the neutron source for the Tank Model. Two methods for generating the neutron source for the Tank Model are presented in Section 3.5.

3.2.2 Tank Models: TORT and MCNP Models

Preliminary calculations show that the execution of MCNP with the core model consumes extremely longer computational time, in the range of days. Therefore, a study was designed to analyze the execution time of the MCNP Core Model to determine how much time MCNP would require for each components, i.e. the core, and the D$_2$O tank, in the simulation. In other words, the time spent transporting particles in each component of the MCNP Core Model was approximately measured and compared. The results of this study show that MCNP5 spends almost 95% of its simulation time for the core calculation and 5% of its simulation time for the transport calculation in the moderator tank in the MCNP Core Model. The gain in the computer time might be by a factor of 20 if the core part is properly removed.
from the MCNP Core Model by defining a boundary neutron source at the core-tank interface to represent the core neutron flows from the core to the tank. In this way, a model named Tank Model is generated with a proper boundary neutron source.

Two- and three-dimensional and 3D views of the Tank Model are presented in Fig. 3.6 and Fig. 3.7, respectively. This model only includes the drum shaped D$_2$O moderator tank filled with D$_2$O, a graphite block behind the moderator tank, a section of beam tube #4 in the moderator tank, termed ”the beam re-entry gap” and the Bi gamma shielding disk. The mesh structure for the model geometry is created using GGTM, then TORT and MCNP inputs are generated by gentinp-3. In the Tank Model a boundary source, which is discussed in Section 3.5, is provided at the PSBR core - D$_2$O moderator tank interface by running a code sequence, which is summarized in Section 3.5 developed to generate the boundary sources for both the TORT and MCNP Tank Models.

The results of the Tank Model for both codes, and their performance analysis are presented in Section 3.7.

![Figure 3.6](image)

**Figure 3.6.** Top-view of the simplified configuration of the beam port of PSBR.
3.2.3 Beam Tube Models: TORT and MCNP Beam Tube Models

The beam tube model developed to compute the neutron output at the beam tube exit end is totally separated from the Core Model and the Tank Model. This model only includes the beam tube without the re-entry gap and the Bi disk, i.e. the exiting face of the Bi disk sits on one of the outer boundaries of this model. This model is computationally very long because it includes the entire length of the beam tube, over 3 m. A schematic of the model is shown in Fig. 3.8.

A boundary neutron source at the entrance of the beam tube is provided to this model by the Tank Model calculations. This source includes the neutrons’ angular, spatial and energy characteristics at the exiting surface of the Bi disk in the Tank Model (at the entrance of the beam tube in the beam tube model). With
this boundary source, the beam tube model is used to compute the neutron output at the beam tube exit end by either TORT or MCNP.

Both TORT and MCNP beam tube models have some disadvantages:

1. MCNP execution without biasing produces inaccurate results at the beam tube exit, and its computation time is too long.

2. TORT beam tube model requires very fine angular quadratures to avoid Ray-Effects (Ray-Effects arise from the approximation of the angular distribution of the particles by a set of discrete ordinates [12]) in the results. The disadvantage of employing very fine angular quadratures instead of coarser angular quadratures is increasing TORT memory requirement and to some extent the computation time.

In order to obtain more accurate estimates of the neutron beam intensity at the beam tube exit end in a reasonable time, a streaming operator was developed.
to avoid performing a transport calculation recognizing the almost non-scattering nature of the air inside the beam tube. Results obtained with this streaming operator and the performance comparison of the beam tube models and the streaming operator are given in Section 3.6.

3.3 Cross-section Preparation for the TORT Models

For the validation of the TORT model configurations a multi-group cross-section library was constructed from ORNL’s ANSL-V Cross-section library. The DOORS/GIP [48] and GipGUI [49] codes were used to prepare the model problem material cross-section data sets in a suitable format for the TORT code.

3.3.1 ANSL-V Cross-section Library

The cross-section data for the model problem were mixed from ANSL-V, an ENDF/B-V Based Multi-group Cross Section Library originally developed for Advanced Neutron Source(ANS) Reactor Studies [50]. The library is available in 99 neutron/44 gamma groups and also in a broad group, 39 neutron/44 gamma, energy structure, with $P_5$ anisotropic scattering representation. The energy group structure of the 39 group neutron library is shown in Table 3.1. In this library the first thermal group is the 15th group.

3.3.1.1 Library Contents

The ANSL-V library includes many reaction cross-sections for neutrons and gammas. These were prepared by using different weight functions at different temperatures, and then they were expanded in a set of Legendre coefficients. A special procedure was developed to calculate Ortho- and Para- Hydrogen and Deuterium (using very low temperature kinematics) at cryogenic temperatures in order to be able to use data in cold source modeling activities.
Table 3.1. Neutron energy upper boundaries for the 39-group ANSL-V library(eV)

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0000E+07</td>
<td>11</td>
<td>5.5000E+02</td>
<td>21</td>
<td>4.7900E-01</td>
<td>31</td>
<td>4.4500E-03</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6.4340E+06</td>
<td>12</td>
<td>1.0000E+02</td>
<td>22</td>
<td>3.9700E-01</td>
<td>32</td>
<td>3.2500E-03</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.0000E+06</td>
<td>13</td>
<td>3.0000E+01</td>
<td>23</td>
<td>3.3000E-01</td>
<td>33</td>
<td>2.6000E-03</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.8500E+06</td>
<td>14</td>
<td>1.0000E+01</td>
<td>24</td>
<td>2.7000E-01</td>
<td>34</td>
<td>2.1500E-03</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.4000E+06</td>
<td>15</td>
<td>3.0000E+00</td>
<td>25</td>
<td>2.1500E-01</td>
<td>35</td>
<td>1.8000E-03</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>9.0000E+05</td>
<td>16</td>
<td>1.7700E+00</td>
<td>26</td>
<td>1.6200E-01</td>
<td>36</td>
<td>1.4500E-03</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4.0000E+05</td>
<td>17</td>
<td>1.3000E+00</td>
<td>27</td>
<td>1.0400E-01</td>
<td>37</td>
<td>1.1500E-03</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.0000E+05</td>
<td>18</td>
<td>1.0000E+00</td>
<td>28</td>
<td>5.0000E-02</td>
<td>38</td>
<td>8.5000E-04</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1.7000E+04</td>
<td>19</td>
<td>7.6500E-01</td>
<td>29</td>
<td>3.0000E-02</td>
<td>39</td>
<td>5.5000E-04</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3.0000E+03</td>
<td>20</td>
<td>6.2500E-01</td>
<td>30</td>
<td>1.0000E-02</td>
<td>40</td>
<td>1.0000E-05</td>
<td></td>
</tr>
</tbody>
</table>

3.3.2 Cross-section Preparation Module for TORT

The ANSL-V cross-section library was prepared in BCD (Binary Coded Decimal, each digit is represented by four bits) format. ORNL’s SCALE code system has a variety of cross-section processing tools comprising the AMPX package[51]. By using AMPX modules, this library can be processed and can be converted to an AMPX master library format (binary format) or working library format (machine dependent binary format).

The GIP program prepares cross-sections for the various DOORS codes including TORT. It uses cross-section libraries in ANISN/DOT3 format. In order to obtain ANISN/DOT3 format from BCD format, a converter program is necessary.

Although one of the AMPX modules is capable of converting any library format to ANISN/DOT3 library format, the results of this module are not arranged in suitable order. Thus, originally a set of programs were developed and used to convert the AMPX master library format to ANISN/DOT3 library format. The cross-section generation code sequence, see flow chart in Fig. 3.9, was designed to prepare the necessary cross-sections for the neutronics models. The functions of each module in this code sequence are:

1. **INPUT**: ANSL-V library in BCD format.

2. **AIM (AMPX)**: Converts BCD library format to AMPX master library format.
3. **PALEALE (AMPX)**: Detailed description of master library (user can specify and select isotopes, temperatures, reaction types, etc.).

4. **KBB-CONVERTER**: Converts master library or working library to ANISN/DOT3 library (user specify and selects isotopes, temperatures, reaction types, etc.)

![Diagram](image.png)

**Figure 3.9.** A code sequence to prepare cross section in ANISN/DOT3 format

5. **VERIFIER**: Compares ANISN/DOT3 format and MASTER library format (user selection)
6. **KBB-COLLAPS**: Collapses the 39 group library to a 26 group library using the spectrum provided by MCNP full model calculation. The collapsed neutron library has one fast group and one epithermal group. The first thermal group in the collapsed library becomes Group 3.

7. **GIPGUI**: A graphical user interface for GIP input preparation. User can easily specify mixtures and material cross-sections by using this program [49].

8. **GIP**: Cross-section preparation program for TORT.

Subsequently, a second cross-section preparation sequence for which the flow chart is given in Fig. 3.10 was also developed after resolving the results of the AMPX module. This sequence was used to test the results of the previous sequence presented above. In this sequence, cross-section conversion and group collapsing processes were performed via AMPX routines, and cross-section mixing was done via GipGUI. The functions of each module in the second sequence are:

1. **INPUT**: ANSL-V library in BCD format.

2. **AIM (AMPX)**: Converts BCD library format to AMPX master library format.

3. **AJAX (AMPX)**: Obtains the reduced library which includes the selected isotopes from the master library.

4. **MALOCS (AMPX)**: Collapses the 39 group library to a 26 group library using the spectrum provided by the MCNP Core Model calculation. The collapsed neutron library has one fast group and one epithermal group. The first thermal group in the collapsed library becomes Group 3.

5. **AJAX (AMPX)**: Converts the master library to ANISN/DOT3 library.

6. **GIPGUI**: A graphical user interface for GIP input preparation. User can easily specify mixtures and material cross-sections by using this program.

7. **GIP**: Cross-section preparation program for TORT.

These two code sequences generate both 39 group neutron and 26 group neutron libraries for the TORT calculation with the developed neutronic models.
3.3.3 Sample Cross Section Library for the Model Problems

For the validation of the physics code TORT and the TORT model problems a cross-section library including a few nuclides was constructed using the code sequence for the cross-section preparation. Properties of this library are:

1. Includes H-bound, D-bound, C, C-graphite, N, O, Al, Bi,
2. Thermal scattering is at 300, 400, 500, 700, 900 K

3. Maximum 5th order Legendre expansion of anisotropic scattering

4. Includes total, absorption, elastic and inelastic scattering, \((n,\gamma)\) reaction cross-sections

5. 39 neutron groups.

For the model problems, a 26 group library was prepared in ANISN/DOT3 format by collapsing the first 14 groups of the 39 group neutron library into one group via a spectrum computed by MCNP full model. The energy range of the first group is from 3.0 eV to 20.0 MeV and for the remaining groups the energy varies from 0.00001 eV to 3.0 eV. Then using the GIPGUI program, cross-section data for \(H_2O\), \(D_2O\) and air (lower density dry air) were produced.

### 3.3.4 Testing Generated Cross-section Library with TORT

In order to test the generated cross-sections, a simple geometry shown in Fig. 3.11 was modeled with both TORT and MCNP and their results were compared for the homogeneous blocks of different materials. In the model a boundary source was defined at the front face of the block (right boundary) and the results, the outgoing partial currents, were computed at the top boundary \((z=20\) plane) by both codes. The same calculation was repeated using water, heavy water, graphite, aluminum, air and bismuth.

In addition, solutions computed by TORT using cross-sections including only \(P_0\) isotropic scattering and including higher order scattering were compared. The MCNP calculations were based on continuous energy cross-sections, with the outgoing flux on the top face shown in Fig. 3.11 tallied over the 26-group structure employed in the multi-group cross-section library.

The results are presented in Figs. 3.12-3.16 obtained for the various materials listed above. The results show that the outputs of TORT with the 26 group-collapsed ANSL-V library are in excellent agreement with the outputs of the MCNP5 continuous energy calculation over most of the energy range. The largest discrepancy between the results of these codes occurs in Group 1, the fast group, for all materials. This result is not surprising since this group was defined over a
Figure 3.11. Configuration prepared for testing generated cross-sections

very broad energy range (3eV to 20MeV) in the ANSL-V library, and TORT uses as single value for the neutron cross-section in this energy range (neutron cross-section averaged over this interval, 3eV to 20 MeV, to obtain cross-sections for this group) whereas MCNP simulates neutron interactions in this range by using continuous energy treatment. In other words, in this broad energy range, MCNP treats neutron interactions in more detail than TORT.

Although a large disagreement appears in the first group, the 26 group collapsed ANSL-V library can be used with the TORT code for the beam port model of PSBR since the PSBR core is over-moderated due to the presence of ZrH in fuel element, and the neutron source spectrum near the core interface is well thermalized. The source spectrum at that interface is presented in Section 3.5.

The most important conclusion reached from the comparisons is that using the ANSL-V cross-sections with TORT is suitable for the PSBR beam port optimization calculations. Moreover, in the optimization calculation only cross-sections including $P_0$ scattering need to be used since the results of $P_0$ order and $P_3$ order agree well for different materials. The advantage of employing $P_0$ instead of a higher order scattering is reducing TORT’s memory requirement and to same extent computation time.
Figure 3.12. Results computed with MCNP and TORT models for homogeneous water block

Figure 3.13. Results computed with MCNP and TORT models for homogeneous heavy water block
Figure 3.14. Results computed with MCNP and TORT models for homogeneous aluminum block

Figure 3.15. Results computed with MCNP and TORT models for homogeneous bismuth block
3.4 Angular Quadrature Selection in TORT

Angular quadrature selection for TORT depends on the problem being solved. Although using higher order quadratures reduces Ray-Effects, it drastically increases computation time. Thus, careful selection of the quadrature is crucial for a successful calculation. Moreover, some problems need biased angular quadrature sets and TORT permits the user to employ a biased set most suitable for their calculations.

3.4.1 Preparing the Angular Quadrature Set

A sub-module was implemented into the input generator gentinp-3 to prepare either predefined angular quadrature sets or user supplied angular quadratures sets in a format suitable for the TORT input file. This sub-module called “xs-ncon3” contains some fully-symmetric and some biased quadrature sets as shown in Table 3.2. When the user enters a quadrature set specification in the input file, this routine checks the data and prepares the requested quadrature set for TORT.
### Table 3.2. Angular Quadrature sets included in sub-module *xsneon3*

<table>
<thead>
<tr>
<th>Symmetrical</th>
<th>Upward-biased</th>
<th>Downward-biased</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_2$, 12 angles</td>
<td>UP-100, 200 angles</td>
<td>DOWN-100, 200 angles</td>
</tr>
<tr>
<td>$S_4$, 32 angles</td>
<td>(130 up / 70 down)</td>
<td>(70 up / 130 down)</td>
</tr>
<tr>
<td>$S_6$, 60 angles</td>
<td>UP-166, 332 angles</td>
<td>DOWN-166, 420 angles</td>
</tr>
<tr>
<td>$S_8$, 96 angles</td>
<td>(262 up / 70 down)</td>
<td>(70 up / 262 down)</td>
</tr>
<tr>
<td>$S_{10}$, 144 angles</td>
<td>UP-210, 420 angles</td>
<td>DOWN-210, 420 angles</td>
</tr>
<tr>
<td>$S_{12}$, 192 angles</td>
<td>(306 up / 114 down)</td>
<td>(114 up / 306 down)</td>
</tr>
<tr>
<td>$S_{14}$, 252 angles</td>
<td>UP-315, 730 angles</td>
<td>DOWN-315, 730 angles</td>
</tr>
<tr>
<td>$S_{16}$, 320 angles</td>
<td>(560 up / 70 down)</td>
<td>(70 up / 560 down)</td>
</tr>
</tbody>
</table>

#### 3.5 TORT Source Module for the Tank Model

In order to simulate the PSBR core neutrons entering the D$_2$O moderator tank for the Tank Model, whose configuration is depicted in Fig. 3.6, a boundary neutron source was defined at the core-moderator tank interface by performing PSBR core calculations in two different ways: (1) MCNP unit-cell/ADMARC-H source calculation [22] and (2) MCNP full model calculation.

This section presents a detailed description and comparison of these sources, and the developed source preparation tools.

#### 3.5.1 MCNP Unit-cell/ADMARC-H Source

The properties of this source distribution are:

1. **26-group energy spectrum:** In earlier PSBR calculations a 12-group neutron source spectrum was estimated by performing a preliminary lattice cell calculation by MCNP on an average reactor cell [22, 28, 29]. In order to compare TORT results to the previous MCNP results obtained by Sarikaya and Alim, this source spectrum was expanded to 26 energy groups coinciding with the cross-section library utilized in the TORT model. The 26-group spectrum of the source neutrons obtained by this approach is given in Fig. 3.17.

2. **Angular distribution:** Assumed the neutrons passing the core-D$_2$O tank interface have semi-isotropic angular distribution, i.e. uniform distribution over all incoming angles.
3. Spatial distribution: Provided by Sarikaya, it was obtained by ADMARC-H, 2-group PSBR core calculation.

![Neutron source spectrum obtained by MCNP unit-cell calculation for both TORT and MCNP computational models](image)

**Figure 3.17.** Neutron source spectrum obtained by MCNP unit-cell calculation for both TORT and MCNP computational models

By using the above prescription, two different sources were prepared. The first one has a spatially uniform distribution, and the second has the spatial distribution computed by ADMARC-H depicted in Fig. 3.18. These distributions were used to prepare the source distribution input of both TORT and MCNP. Figure 3.18 shows that the neutron spatial distribution computed by ADMARC-H is symmetric with respect to \( y = 0 \) plane. In contrast, the neutron axial distribution is not symmetric about the \( z = 0 \) plane since the control rods were inserted in the core in these calculations. However, the vertical spatial distribution has a systematic error. It is a fact that the neutron distribution does not vanish at the point \( z = -14 \text{ cm} \) because this location almost coincides with the bottom edge of the fuel rods and therefore; a significant amount of neutron should pass from the core to the moderator tank. This is also supported by the second source generation approach given in next section.
3.5.2 Source Module: A Realistic Source Generation Approach

In the Tank Model, in order to simulate the core neutrons coming from the core entering the moderator tank through its interface with the reactor core, an interface neutron source was modeled. It was endowed a spatial distribution, an angular distribution, and an energy spectrum computed by performing an MCNP full model calculation that included both the PSBR core and the moderator tank filled entirely with D$_2$O. The layout of the MCNP-full model is given in Fig. 3.19.

For this purpose, a code sequence given in Fig. 3.20 was developed to generate the interface or boundary neutron source for the Tank Model. A source module was developed to profile the neutron data provided by the MCNP Core Model calculations, and then assembled to the code sequence to generate neutron boundary source at the PSBR core-D$_2$O moderator tank interface.

In this sequence, all particles crossing the core-D$_2$O moderator tank interface are scored into a file by using the SSW function of MCNP. Then the developed
The source module uses the following source generation strategy:

1. Initially, an MCNP Core Model calculation is performed with the SSW function enabled.


3. The module calculates the energy, angle and spatial intervals using the provided TORT mesh data and angular quadrature files. These are used to bin the particles’ dependence in the process of constructing the PDF for MCNP or the source arrays for TORT.

4. Normalized distributions and normalization parameters are obtained for both the neutron and gamma source distributions based on the PSBR operating power.

**Figure 3.19.** MCNP full model for source calculation
Figure 3.20. The code sequence for generating a boundary source for the Tank Model

5. The desired source is generated and written to file for:

(a) A boundary source file for TORT Tank Model.

(b) A SSR (Surface Source Read function of MCNP5 [17]) particle information file obtained by re-sampling the particles at the source interface that will be used in the MCNP Tank Model as source input data.

(c) A set of probability density functions represent the spatial, angular, and energy distributions of particles at the given interface. These are useful to define the interface source in the MCNP input file via MCNP’s source definition card, SDEF, for the MCNP Tank Model.

Using the above procedure, two different sources were prepared. One of them has a spatially uniform distribution, and the other has a spatial distribution computed by the source module from the outputs of the MCNP Core Model. The
spectra and the spatial distributions for core neutrons and gammas are given in Figs. 3.21-3.24.

In this way, not only the source distribution was obtained to drive the TORT Tank Model, but also a re-sampling strategy was developed for to drive the MCNP Tank Model calculation. As anticipated, the execution time of the MCNP full model calculation is too long and the tally error for the neutron beam all the way at the beam exit is unacceptably large. Hence, dividing the full model into two or more sections and re-sampling particles from one section’s output to the other’s input are essential for the success of the calculation. After analyzing the neutron source at the interface, it was concluded that neutrons roughly behave semi-isotropically over incoming directions for the existing beam port’s neutronic models. Thus for the TORT verification calculations with these neutronic models, the angular distribution for the generated source was fixed as a semi-isotropic distribution rather than using the angular distribution obtained by the source module.

Figure 3.21. Neutron source spectrum at the interface as computed by the source module.
**Figure 3.22.** Normalized spatial distribution of neutrons obtained by analyzing the MCNP neutron histories crossing the core-D$_2$O tank interface.

**Figure 3.23.** Gamma source spectrum at the interface as computed by the source module.
3.5.3 Validation Studies for the Boundary Source

A case study was designed to test the two different source generation approaches described above by performing both TORT and MCNP calculations on the Tank Models.

The spectrum of MCNP-computed neutron source was binned in 26 groups and compared with the energy distribution of the source obtained by the MCNP unit-cell/ADMARC-H calculation. The comparison of these two spectra is provided in Fig. 3.25. The source generated by the MCNP unit cell calculation gives higher neutron fluxes in the thermal and higher energy region than the source generated by the MCNP full model calculation.

The following four different neutron sources were generated to simulate the core neutron source at the entrance of the Tank Models introduced in Section 3.2.2:

1. **Source1-Uniform**: MCNP unit-cell spectrum, uniform spatial distribution, semi-isotropic source.

2. **Source1-Distributed**: MCNP unit-cell spectrum, spatial distribution by
Figure 3.25. Comparison of two source spectra computed by the two different approaches described in Section 3.5.2

ADMARC-H, semi-isotropic source.


4. Source2-Distributed: MCNP full-model spectrum, spatial distribution by MCNP full-model, semi-isotropic source.

TORT and MCNP calculations were performed with these four source distributions using the current design of the D$_2$O Tank Model which is depicted in Figs. 3.6 and 3.7. The outgoing flux and current were computed at the exit face of the bismuth disk, point A in Figs. 3.6 and 3.7. In thermal and lower energy groups, TORT with both sources produces similar neutron partial currents flux at the exit surface of the bismuth disk as presented in Fig. 3.26.

In order to test the effect of the spatial distribution of the source on the neutron partial current at the exit surface of the bismuth disk, the same neutron spectrum was used with the uniform and spatially distributed neutron sources. The comparison is given in Fig. 3.27. The figure shows that the spatial distribution does
Comparison of TORT results after bismuth

Figure 3.26. Comparison of partial currents exiting the Bi disk produced by TORT with a spatially uniform source and two different spectra.

Figure 3.27. Comparison of partial currents exiting the Bi disk produced by TORT with two different spatial distributions.
not have a strong effect on the spectrum of the partial current at the exit face of the bismuth disk since the D$_2$O material smoothes out the spatial non-uniformities of the source neutrons as they traverse it. While this may be true for the specified model problem (existing beam tube), it might fail if the moderator tank shape and size change drastically during the optimization search. Therefore, fairly detailed source calculations are necessary to maintain the accuracy of the optimization calculations sequence.

Similar calculations described above were performed with MCNP. The neutrons were spatially re-sampled at the D$_2$O moderator tank - core interface by the source module using the TORT mesh structure at that interface. A cell-centered point source was defined for each cell by using the probability that a neutron crosses from the interface within that cell. This probability was calculated as the ratio of the number of neutrons crossing from the interface from within this cell to the total number of neutrons crossing the entire source interface. Then, the computed probabilities were entered into MCNP’s source card as a cell centered point source. The error in the calculation of the probability for the cell’s source was ignored in the re-sampling algorithm. This assumption is reasonable due to the large number of cells in the TORT model, at that interface; there are 2600 cells, each with about 1.05 cm$^2$ in surface area.

The comparison of the two MCNP-computed spectra with uniform spatial distribution is shown in Fig. 3.28 and the effect of the spatial distribution is presented in Fig. 3.29. The former figure, Fig. 3.26, shows a difference at the upper thermal region between the partial currents computed with the two source spectra. In contrast, the spatial distribution of source neutrons is insignificant for accurate modeling of the exiting partial current at the Bi disk in this configuration.

Comparison of the TORT and MCNP results employing a uniform and non-uniform spatial distributions are presented in Figs. 3.30 and 3.31, respectively showing that the two codes produce comparable results in spite of the drastically different solution methods, discrete-ordinates versus stochastic, geometric model, stair-casing versus exact, and nuclear data energy dependence, multi-group versus continuous. The large discrepancy between the TORT and MCNP results in the first 3 groups ( $E \geq 1$eV ) is primarily due to poor statistics of MCNP simulations in these 3 groups ($\geq 50\%$).
Figure 3.28. Comparison of partial currents exiting the Bi disk produced by MCNP with a spatially uniform source and two different spectra.

Figure 3.29. Comparison of partial currents exiting the Bi disk produced by MCNP with two different spatial distributions.
Comparison of MCNP and TORT results after bismuth

Figure 3.30. Comparison of partial currents exiting the Bi disk produced by TORT and MCNP with a source with uniform spatial distribution.

Figure 3.31. Comparison of partial currents exiting the Bi disk produced by TORT and MCNP with a source whose spatial distribution is computed by the source module.
3.6 Streaming Operator

The streaming operator attenuates the angular neutron flux emanating from a surface within the beam tube (e.g., exiting the Bi disk) to the beam exit hole (sometimes called the beam tube exit end). The streaming operator was developed to compute the neutron output at the beam tube exit end without performing a transport computation along the entire length of the beam tube. Such transport computation, whether conducted with Monte Carlo or Discrete Ordinates, was found to incur large errors and consume long execution times.

The calculation procedure of this operator is:

1. A number of pixels (2-D cells) is defined in the plane of the beam tube exit hole.

2. In order to determine the neutron contribution from the surface flux exiting a computational cell on the exit face of the Bi disk to all pixels on the exit hole, the unit surface area and its projection on the beam tube exit are calculated for all angles.

3. A weight, $\alpha_m(i,j) \rightarrow (i_1,j_1)$, is defined to represent the fractional contribution of the neutrons from the cell-surface $C_{i,j}$ to the pixel $P_{i_1,j_1}$ for the angle $\Omega_m$. It is computed as the area of the intersection between the projection of the unit surface area associated with the angle $\Omega_m$ on the exit hole plane and pixel $P_{i_1,j_1}$ surface area. This weight is a function of the surface areas of $C_{i,j}$ and $P_{i_1,j_1}$, angle $\Omega_m$ and the distance between $C_{i,j}$ and $P_{i_1,j_1}$. See Fig. 3.32.

4. By using boundary fluxes denoted as $\psi_{i,j}^m(r)$, outgoing fluxes, $\phi_{g,i_1,j_1}^m$, are calculated at the exit hole.

The outgoing flux for any pixel at the exit hole is calculated as:

$$
\phi_{g,i_1,j_1}^m = \left[ \sum_m^{MM} \omega_m \left( \sum_{i,j} \alpha_m(i,j) \rightarrow (i_1,j_1) \psi_{i,j}^m(r) A_{i,j} \right) \right] e^{-\Sigma_g(z-z_0)} \quad , \quad \eta_m > 0 \quad (3.1)
$$

where $\omega_m$ is the angular weight of the $m^{th}$ discrete ordinate, $\eta_m$ is the angle cosine, and $A_{i,j}$ is the surface area of the cell-surface $C_{i,j}$. The partial current density is:
For any energy group, the total partial current, also termed beam intensity, can be written as:

$$J^+_{g,i_1,j_1} = \left[ \sum_m^{MM} \omega_m \eta_m \left( \sum_{i,j} \alpha^m_{(i,j)\rightarrow(i_1,j_1)} \psi^g_{i,j} (r) A_{i,j} \right) \right] e^{-\Sigma_g (z-z_0)} , \ \eta_m > 0$$

(3.2)

If the surface areas of the pixels at the exit hole are not equal, an area correction should be performed for the outgoing current calculation,

$$J^+_{g} = \sum_{i_1,j_1} \left[ \sum_m^{MM} \omega_m \eta_m \left( \sum_{i,j} \alpha^m_{(i,j)\rightarrow(i_1,j_1)} \psi^g_{i,j} (r) A_{i,j} \right) \right] e^{-\Sigma_g (z-z_0)} , \ \eta_m > 0$$

(3.3)

$$J^+_{g} = \left[ \sum_{i_1,j_1} P_{i_1,j_1} \sum_m^{MM} \omega_m \eta_m \left( \sum_{i,j} \alpha^m_{(i,j)\rightarrow(i_1,j_1)} \psi^g_{i,j} (r) A_{i,j} \right) \right] e^{-\Sigma_g (z-z_0)} , \ \eta_m > 0$$

(3.4)
where $P_{i_1,j_1}$ is the surface area of pixel $(i_1,j_1)$.

### 3.6.1 Verification of the Streaming Calculations

In order to validate the streaming operator, a simple model configuration of the beam tube was designed. The model shown in Fig. 3.33 includes a beam tube ($L=301.2$ cm $\times$ $D=20.0$ cm) and an arrangement of collimators.

A boundary neutron source was specified by using the TORT Tank Model at the beam tube entrance, point B in Figs. 3.6 and 3.7, and the partial current was calculated at the beam tube exit by applying the streaming operator. TORT and MCNP calculations were performed using the same configuration, and results of these codes and of the streaming operator were compared. The results given in Fig. 3.34 indicate that the streaming operator computes the neutron beam spectrum at the exit hole with a good accuracy compared with the TORT and MCNP results. Moreover, it takes only a few minutes to compute these values whereas TORT spends about 122 minutes and MCNP, with 2 billion particle histories, consumes over 20 days. Poor statistics characterize the MCNP results and are responsible for the large discrepancies observed in some groups. Specifically, three groups had zero tallies, and only 4 groups had tallies with error smaller than 8%. In addition, Ray-effects were observed in the results of the TORT beam tube model with upward biased UP-100 quadrature.
Another advantage of the streaming operator is that it generates a detailed pixel-size image of the neutron spatial distribution at the beam tube exit end. In this way, the spatial uniformity of the neutron beam can be determined, and its adequacy for a specific experiment’s requirements can be judged. A sample neutron spatial distribution computed by the streaming operator is shown in Fig. 3.35. 

**Figure 3.34.** Comparison of the outgoing partial currents at the exit hole (computed by the BT models and the streaming operator)
A set of verification results for the physics code, TORT, were given in Section 3.5.3. Both TORT and MCNP calculations were performed on the Tank Model by using the same source configuration. The two different solution methods produced comparable results for the entire energy range below 1 eV. This is due to the poor statistics of the MCNP simulations in this energy range ($E \geq 1$ eV).

In this section, a comparison is performed between the TORT and MCNP models. For the same configuration, the MCNP Core Model, the MCNP and TORT Tank Models were designed and their inputs were generated by `gentinp-3`. After executing MCNP5 with the MCNP Core Model consuming a long computational time, neutron source data were generated at the core-tank interface for the Tank Models. Then, the interface sources for both the TORT and MCNP Tank Models were generated by the source module. Finally, the TORT Tank Model was ex-

**Figure 3.35.** Spatial distribution of neutrons per fission event computed by the streaming operator at the beam tube exit end
cuted with the 26 collapsed neutron cross section library and the boundary source computed by the source module, whereas a continuous energy MCNP simulation was performed on the MCNP Tank Model with the boundary source computed by the source module.

The half scalar fluxes emerging from the exiting surface of the Bi disk computed by the three models are presented in Fig. 3.36. The results verify that the TORT Tank Model with the ANSL-V cross-section library and the boundary source computed by source module is adequate for performing design calculations of the PSBR’s beam port facilities.

![Comparison of the half scalar flux after Bi disk produced by TORT and MCNP Tank Models, and MCNP Core Model](image)

**Figure 3.36.** Comparison of the half scalar flux after Bi disk produced by TORT and MCNP Tank Models, and MCNP Core Model

### 3.7.1 Performance of the Models

One of the primary criteria for a successful computational model is achieving high computational efficiency; that is producing accurate results while consuming small
computational resources, e.g. execution time and memory. Therefore a performance analysis of all three models was conducted to determine their relative computational efficiency. The three models were executed on the same computer (3.6 GHz Intel Pentium-4 EM64 CPU with 1MB L2 cache memory, 2 GB physical memory, 15000 RPM Ultra-3 SCSI hard drive) and their performance was evaluated. The following results were obtained:

1. MCNP Core Model

   (a) computation time is 12,281 minutes = 8.5 days
   (b) 100 million particle histories
   (c) Error is less than 5% in thermal groups
   (d) Poor statistics in other groups (insufficient scoring, error is almost 100%)
   (e) The number of neutrons crossing the D$_2$O tank - core interface and scoring after Bi is 1,820,213 and 71,338, respectively. In other words, the contribution of one neutron generated by fission in the core to the neutron scoring on the exit face of the Bi disk is only 0.071%. As a result, it is easily concluded that the comprehensive model including the core, the tank and the beam tube is not feasible for computing the neutron output at the beam tube exit end.

2. MCNP Tank Model

   (a) Started with a boundary source (SSR, re-sampled neutrons by source module)
   (b) computation time is 1,508 minutes = 25.2 hours
   (c) 100 million particle histories
   (d) Error is less than 3% in thermal groups
   (e) Poor statistics in fast and epithermal groups (insufficient scoring, error is greater than 50%)
3. TORT Tank Model

(a) computation time is 570 minutes = 9.5 hours for fine mesh calculation
(36×43×51 Cartesian mesh, UPWARD-100 biased quadrature and 26 group cross-sections collapsed from the ANSL-V library with isotropic scattering representation)

(b) computation time is 31 minutes = 0.5 on a coarse mesh (21×26×22)

(c) The maximum difference (over space and energy) in the resulting outgoing partial current computed by TORT on the fine and coarse meshes is 13%.

The results show that the shorter computation time enables usage of TORT as a transport solver in the physics code for the modular optimization code package to design the new beam ports for the PSBR in a reliable and accurate way. Because of the poor statistics of the tallied result, using MCNP without biasing techniques is not suitable for our purposes. Proficient use of biasing techniques requires substantial expertise and some knowledge of the solution’s distribution over phase space.

3.8 Summary of the Computational Models

To summarize the results given in this chapter, it is concluded that:

1. TORT is suitable for the modular optimization sequence because of its shorter execution time for on a given model.

2. MCNP without biasing is not practical to use for the modular optimization sequence.

3. ANSL-V cross-section set is suitable for the PSBR’s beam tubes design calculations.

4. TORT with very fine angular quadrature computes the desired quantities without significant Ray-Effects.
5. It is possible to recalculate the boundary source during the optimization sequence.

6. The streaming operator accurately computes the neutron beam intensity at the exit end of the beam tube without performing a transport calculations along the beam tube’s length in a few minutes.

7. The TORT Tank Model with the computed boundary source and streaming operator work well together to compute the fluxes/currents at the beam tube exit end (reasonable computation times of approximately 30 min and 2 min, respectively).

8. MCNP models can be used for the validation calculations performed for the optimal configuration at the conclusion of the optimization sequence.
Chapter 4

Analysis Performed to Test MOZAIK

Most code developers test their code systems by designing some computational experiments or numerical benchmarks before their code is frozen and then released to be used in real applications. In this way, the functionality of the code system can be tested in its entirety and any uncovered bugs can be fixed. For this purpose, an optimization problem using the current beam port configuration of the Penn State Breazeale Reactor was designed to test the functionality of MOZAIK and its independent modules before starting the actual optimization calculation.

In this chapter, a detailed description of the selected test problem, preparation of MOZAIK’s modules to apply to this problem, and results of MOZAIK’s performance analysis for this specific model problem are presented.

4.1 Testing the Optimization Sequence

In order to test the entire code package, a sample case was modeled based on the PSBR existing beam port facility. The objective of this test case is to determine the optimal shape of the D$_2$O moderator tank which maximizes the thermal neutron beam intensity exiting at the beam tube for the simplified model of PSBR described in Chapter 3. For this purpose, a simplified neutronic model was obtained by modifying the PSBR’s Tank Model. Then, the search problem domain was constructed and a suitable cost function was defined. Finally, the optimization
calculations were performed by MOZAIK employing TORT in the physics module and Min-max with breadth-first search with pruning in the optimizer module.

4.1.1 Neutronic Model Development

The following assumptions were made to simplify the neutronic model for the optimization problem:

1. The size and location of the beam tube was fixed:
   The geometry, i.e. the shape and dimensions, of the beam tube re-entry hole was fixed and an optimal beam tube location relative to core-D$_2$O tank interface was manually sought over the problem domain. For this purpose a few variations of the model inputs were prepared by moving the beam tube from its original location (H=19.14 cm as depicted in Fig. 4.1) to positions closer to the core interface (H=7.62 cm). In this way, the best location among the attempted positions for the beam tube in the D$_2$O tank was determined. The computed partial current spectra for the various attempted locations are presented in Fig. 4.2. The location of the beam tube that produces the highest neutron partial current output was found to be when it sits immediately adjacent to the core interface.

2. All aluminum components were removed from the Tank Model; these include the moderator drum material, Al around the beam re-entry gap, etc. In addition the graphite block behind the D$_2$O moderator tank was also removed from the model. The simplified model without these components is shown in Fig 4.1-c.

3. The boundary source calculation for the Tank Model presented in Sections 3.5.2 and 3.5.3 shows that the neutron spatial distribution is not axially symmetric because of the control rods. However, in order to speed up MOZAIK for purpose of this test, the neutron spatial distribution at the core-tank interface was assumed to be exactly symmetric. Therefore, for the simplified configuration the neutronic model was axially divided into two pieces and a reflective boundary condition was applied at the symmetry plane in TORT’s input thus allowing solution of only half the problem extent. This reduces
Figure 4.1. Top-view of the PSBR’s Tank Models: (a) Tank Model, (b) Tank Model with optimal beam tube location, (c) Simplified Tank Model for the optimization study.

the number of computational cells by a factor of two for both the physics and
the optimizer modules. As a result of this, the computation time consumed
by the physics code for each state decreases and the optimization calculation
is completed faster than if the source asymmetry was retained. The top-view
of this model from the symmetry plane, and the front-view also are shown
4. Preliminary calculations show that utilizing very fine angular discretization clustered around the beam tube axis deters Ray-effects in the flux distribution within beam re-entry hole and also increases the accuracy of the streaming operator. Therefore, an upward biased quadrature set was prepared for testing MOZAIK’s performance. The model configuration for TORT was rotated to make the beam tube axis coincide with the z-axis about which the upward-biased angles are clustered.

5. During the optimization process, it was assumed that the material distribution in the problem domain did not affect the neutron boundary source generated by the source module. Therefore, an initial source distribution was generated by the source module based on the initial configuration and
Figure 4.3. Top and front view of the symmetric model designed for the testing MOZAIK’s performance.

the physics code TORT used this boundary source for all states during the optimization calculation.

4.1.2 Preparing the Physics Module

The physics module consists of three major sub-modules: input preparation, execution and output processing. The input preparation and output processing depend on the physics code. For the transport problem at hand, the input preparation and output post-processing sub-modules for TORT were tested.

Basically, the input preparation activity generates the material boundaries and the material dependent data sections of the input file(s) for the physics code. In order to perform this task, the input preparation part uses the material distribution of the state under evaluation that has been generated by either the optimizer module or the initializer module.

The output processing activity also depends on the physics code and the op-
timization problem being solved. In the model configuration, although the search problem domain does not include the whole beam tube, the cost function needs the neutron partial currents at the beam tube exit. Therefore, the streaming operator described in Section 3.6 is used to compute the neutron beam intensity at the beam tube exit from each state’s computed flux at the Bi disk exit face.

4.1.2.1 Fixing TORT problem parameters

Although some of the TORT parameters are generated automatically by the initializer and pre-processor modules, others must be entered as input or in an input template before the calculation starts.

The input preparation module sets the following parameters automatically:

1. I/O logical units (linked to files): the cross-section unit, the boundary source input unit\(^1\), distributed source input unit\(^1\), and the flux output unit\(^1\)
2. Maximum order of the flux angular expansion \(^2\)
3. Maximum number of x,y,z spatial mesh intervals
4. Coordinate system is x-y-z (i.e., Cartesian)
5. Number of material zones\(^3\)
6. Number of material zone bodies\(^3\)
7. Necessary patch data for the boundary flux and cell averaged scalar flux

The following input parameters were fixed in the TORT input template;

1. Solution method, (\(\theta\)-weighted with \(\theta=0.9\)).
3. Maximum number of outer and inner iterations.
4. Maximum number of directions in the angular quadrature set.

\(^1\)if these are necessary for the calculation
\(^2\)if it is not set in the input file, the default value is the maximum order of the scattering expansion in the cross-section library
\(^3\)calculated in geometry preparation

6. Source iteration convergence criteria.

7. All cross-section control parameters consistent with the cross-section data file.

8. Initial memory allocation.


Although the last two items can be set automatically by the hardware-checker sub-module in the initializer module, they were set manually in the preliminary calculations reported here. The maximum inner and outer iterations were set to 80 and 20, respectively, and the convergence criteria for the inner and outer iterations were set to $10^{-4}$ and $10^{-3}$, respectively.

### 4.1.3 Model Development for Optimization

In this preliminary test study, Min-max with the breadth-first search algorithm was selected as the optimization algorithm for the optimizer module. Each state’s physics calculation was performed by TORT followed by the streaming operator.

Initially, a search problem domain was defined for the optimization problem by:

1. Removing the D$_2$O moderator from the system and filling the problem domain with water.

2. Placing the beam tube at the location obtained via manual optimization in the above described preliminary optimization calculation.

3. Placing the Bi gamma shielding disk at the junction between the beam re-entry gap and the beam tube.

4. Defining the problem domain on a 16×20×25 Cartesian grid, i.e. 8000 computational cells.
In the first step, a number of states, i.e. new material configurations, are defined by replacing the material assignment of individual water-filled computational cells with D$_2$O one cell at a time for all cells that share a boundary with. One of the initial states is presented by way of example in Fig. 4.4. In the next step, all cells neighboring a D$_2$O cell are checked one by one. If the checked cell contains H$_2$O, it is replaced with D$_2$O and a new state is generated.

![Diagram of material distribution](image)

**Figure 4.4.** One of the initial state, material distribution (at y=0.0 cm)

### 4.1.4 Optimizer Module

The optimizer module consists of three sub-modules: cost calculation, cost evaluation, and new state generation. The goal sought by the optimization search determines the mathematical formula of the cost function and the operator of the optimization algorithm.

In the preliminary testing of MOZAIK, the Min-Max algorithm was selected as the optimization technique and a simple cost function was defined drive the optimization procedure. Additional advanced requirements on the spectrum’s detail and spatial uniformity of the beam were not considered at this early testing stage.
4.1.4.1 Cost Calculations

In the TORT Tank Model described in Section 3.2.2 the outgoing current can be computed on the exit surface of the Bi disk, surface A in Fig. 4.1. In contrast, the physics module can only model the beam port with re-entry hole up to the Bi disk. It cannot efficiently and accurately perform the transport calculation along the 3 meter air-filled beam tube. Therefore, two different approaches were used:

1. At the outer surface of the Bi disk, the angular directions were selected as close to parallel to the beam tube’s axis as possible. This way, the contribution of the neutrons emerging from the Bi disk to the beam intensity at the exit hole was computed with some approximations, then the exiting thermal beam intensity was used for the cost calculation.

2. A streaming operator was implemented, as described in Section 3.6, to attenuate the neutron flux along the beam tube’s length. The so-computed angular and scalar fluxes at the beam tube’s exit hole are then used in computing the cost calculation. This methodology is illustrated in Fig. 4.5.

Very often, experimental users of the PSBR beam facility desire a high thermal neutron flux intensity at the exit hole. In the calculations presented in this study, 0.1 eV was selected the thermal cutoff energy (\(E_{g'}\)) and the contribution to the cost value by neutrons possessing higher energy than this value was de-emphasized in the mathematical formula for the cost function while the contribution of neutrons having lower energy than \(E_{g'}\) was emphasized.

The following formula for the cost function was defined for both approaches,

\[
C_m = -c_1 \times \sum_{g, E_g < E_{g'}} \sum_{i,j} J_{i,j,m}^g + c_2 \times \sum_{g, E_g > E_{g'}} \sum_{i,j} J_{i,j,m}^g
\]

(4.1)

where \(C_m\) is the cost value of state \(m\); \(c_1\) is a large scalar multiplier to ensure dominance of the lower energies on the cost value; \(c_2\) is a small scalar multiplier to reduce the effect of higher energies on the cost value; and \(J_{i,j,m}^g\) is the \(g^{th}\) group partial current averaged over the surface area of pixel \(i,j\) (see Section 3.6) of the beam tube’s exit hole computed for state \(m\).
For the first approach, the surface averaged partial currents were computed at the exit face of the Bi disk by using the directional fluxes nearly parallel to the axis of the beam tube. Then, the surface averaged partial currents were computed using these values and some approximations at the beam tube exit hole. In the second approach, the surface averaged partial currents were computed directly at the beam tube exit hole by the streaming operator.

4.1.4.2 Cost Evaluation

Convergence of the optimization iterations is checked in this module. The best cost computed for the current iteration is compared to the best cost value of the previous iteration. If the cost value of any state in the current iteration is less than the best value of the previous iteration, this sub-module records all states’ data into a file and sends the best state data with a stop signal to the main module.
4.1.4.3 New State Generation

The same mesh structure defined for the neutronic model was used in the optimization calculation. In other words, MOZAIK’s dual mesh support was not enabled in this performance testing stage. Therefore, both the physics and the optimizer modules used the same mesh, $16 \times 20 \times 25$.

The following operators were defined for Min-max to allow the search algorithm to switch the material content of a computational cell:

1. H$_2$O-D$_2$O replacement operator:
   This operator switches the material assignment of specified computational cell(s) from H$_2$O to D$_2$O (effectively adding D$_2$O into the system). It can operate either as cell-by-cell replacement or as cluster-by-cluster replacement, where a cluster is a pre-specified group of cells.

2. Air-D$_2$O replacement operator:
   This operator switches the material assignment of specified computational cell(s) from air to (effectively adding D$_2$O into the system and enabling the D$_2$O to drift into the re-entry hole). Again, this replacement can be applied on a cell-by-cell replacement or as cluster-by-cluster replacement.

After testing each operator on a simple geometric model, these replacement operators were assembled into the Min-max routine within the optimizer module. The cell-by-cell replacement was designated as the default option for the Min-max algorithm. The cluster-by-cluster replacement was installed as an optional strategy designed to speed up the progress of the search procedure.

In Min-max, as the depth of the search increases, the number of newly generated states increases rapidly. This increases the computation time tremendously since a complete physics calculation is performed for each new state. In order to obtain an optimal shape in a limited amount of time, a pruning operator was also generated for the Min-max. This operator selects the top ten or fewer states in terms of their cost values by using the sorted cost list computed by the cost function calculation and the evaluation routine. This way, the algorithm is forced to search new states around the best performing states identified in each search iteration. In other words, the search space can be localized to a tight neighborhood around the best
states in each depth level in order to reduce the number of new states generated, subsequently reducing the computation time of the physics calculation.

After testing the above described tools in Min-max, the material distribution of the model configuration was used to generate an initial state by placing only H$_2$O cells around the beam re-entry hole. Then, the physics calculation was performed for this state. After processing the output of the physics module for this state, necessary data were written to a file and the optimization algorithm was initiated.

The optimization algorithm reads the data for all states in the previous iteration, computes the cost values in terms of the given cost function, then sorts the states in descending order of the magnitude of their cost values. The new state generation routine picks up only the top 10 states or fewer from the sorted list, and generates and writes each possible new state to a file. Then these data are sent to the physics module to compute the beam intensity for each of the generated states.

### 4.1.5 A Strategy to Reduce TORT Computation Time During the Optimization Procedure

The computational time consumed by an iterative method depends on the number of iterations required to achieve convergence. One of the methods to reduce the number of iterations is supplying a good initial guess for the given problem. Since TORT is able to start a transport calculation from a flux initial guess supplied in the input file, this study benefited from this capability to reduce the number of inner iterations TORT executes for each visited state.

In order to achieve this goal the following methodology is developed:

1. In the first search iteration TORT execution is performed for all initial states.

2. For the next iteration, after determining the best state in the previous iteration of the optimization calculation, the cell-averaged fluxes of this state (for all cells and all energy groups) are written to a file by TORT.

3. Since the new states for the new iterations are evolved from the best state of the previous iteration, the differences between the material distribution of each new state and the material distribution of the best state of the previous
iteration are small. Specifically, the two sets of material distributions will differ in only one cell or cluster. Therefore, the number of iterations for the TORT calculation of the new iteration’s states would be expected to be drastically reduced, if TORT’s transport calculation commences with the flux guess set to the converged solution from previous iteration.

4. Items 2 and 3 are repeated for subsequent search iterations.

In order to test this methodology, the model configuration developed for the cross section validation studies (see Fig. 3.11) was used. In the model a boundary source was defined at the front face of the block (right boundary) and the results, the outgoing fluxes, were computed at the top boundary (z=20 plane) by both codes. The model configuration has the following features:

1. 20×20×20 mesh was generated for TORT (8000 cells).
2. All cells were filled with H₂O (initial state, iteration 0, state-0).
3. A boundary neutron source was defined at one face of the model configuration (fixed neutron source, not affected by the material dependency).
4. 26 group collapsed neutron cross-section library and upward biased 100 quadrature set were used with TORT.
5. The outgoing half scalar flux was computed by TORT for the state-0 in 37 minutes (71 outer/8 inner iterations). Upon convergence of the iterations, the cell-averaged scalar fluxes are written to a file (for use as flux guess in subsequent calculations).
6. Four new states are generated by replacing randomly selected H₂O cells with D₂O cells (only four cells for one state),
7. The outgoing half scalar fluxes were computed for all four states by TORT without initial guess. Then the same calculations were repeated by using as initial guess the converged scalar fluxes from the previous iteration (i.e., the model with only H₂O material assignment).
The results given in Table 4.1 illustrate that using the flux guess input file with TORT reduces the computation time by about a factor of 4 for this model configuration. As a result of this successful outcome, the same methodology was applied to the modular optimization code package to reduce the computation time of the physics module.

**Table 4.1.** Comparison of TORT calculations with and without the flux initial guess set to the all H\(_2\)O converged solution

<table>
<thead>
<tr>
<th>State Number</th>
<th>TORT without Flux Guess</th>
<th>TORT with Flux Guess</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inner</td>
<td>Outer</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>71</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>70</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>71</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>71</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>70</td>
</tr>
</tbody>
</table>

### 4.1.6 Numerical Results

After preparing all necessary data for the optimization problem, MOZAIK was executed separately using the two approaches for computing the cost function as described in Section 4.1.4.1. The optimization results obtained by MOZAIK with the Min-max optimization algorithm are summarized in the next two sections.

#### 4.1.6.1 MOZAIK with Min-max.1: Without Streaming Calculation

In the first approach (denoted here as Min-max.1 for brevity), the cost function was computed by using the exiting partial current from the Bi disk. Only the D\(_2\)O-H\(_2\)O replacement operator was activated with the cluster replacement strategy (8 cells in a cluster) to reduce the total computation time. In addition the pruning operator was enabled to select the 6 best states for each depth-level, i.e. highest cost function values, to reduce the execution time. The Min-max algorithm converges after 87 search iterations and a new D\(_2\)O tank shape is produced for the given model...
problem with the cost function specified by Eq. 4.1 with $c_2 = 100$, $c_1 = 1.e8$ and angles satisfying $\eta_m > 0.99$.

![Variation of cost value in MINMAX algorithm](image)

**Figure 4.6.** Improvement in cost value by Min-max.1 in 87 search iterations

The improvement in the cost value is illustrated in Fig. 4.6. This improvement was measured by comparing the cost value of the best state in each search iteration to the cost value of the initial state. The produced optimal shape is shown in Figs. 4.7-4.10. The resulting moderator tank has almost a hemi-spherical shape with a radius of about R=24 cm.

The performance comparison of the optimal shape with the actual D$_2$O tank currently in operation at the PSBR show that the optimal shape produces a larger partial current at the exit face of the Bi disk and at the beam exit hole. These comparisons are shown in Figs. 4.11 and 4.12, respectively.
Figure 4.7. Optimal D$_2$O tank shape with Min-max-1.

Figure 4.8. Top view of the optimal D$_2$O tank shape with Min-max-1, not showing H$_2$O.

Figure 4.9. Back view of the optimal D$_2$O tank shape with Min-max-1, not showing H$_2$O.
Figure 4.10. Front view of the optimal D$_2$O tank shape with Min-max-1, not showing H$_2$O.

Figure 4.11. Comparison of outgoing current spectra after the Bi disk for the real and optimal D$_2$O tank shapes obtained with Min-max.1
Figure 4.12. Comparison of outgoing current spectra at the exit hole for the real and optimal D$_2$O tank shapes obtained with Min-max.1

4.1.6.2 MOZAIK with Min-max.2: With Streaming Calculation

In the second approach (denoted here Min-max.2 for brevity), the cost function was computed using the streaming operator to determine the beam intensity at the exit hole. Both the D$_2$O-H$_2$O replacement operator and the D$_2$O-Air replacement operator were activated with the cluster replacement strategy (8 cells in a cluster). The D$_2$O-Air replacement operator allows some of the air cells to be replaced with D$_2$O, thus allowing the D$_2$O cells to drift into the beam re-entry hole. For this calculation, the pruning operator was also enabled to select the 10 best states in every depth-level as judged by their cost value. The Min-max algorithm converges after 97 search iterations producing a new D$_2$O moderator shape for the preliminary cost function specified in Eq. 4.1 with $c_2 = 100$, $c_1 = 1.e8$.

In this optimization sequence D$_2$O cells drifted into the bottom part of the re-entry hole, hence changing the re-entry hole’s shape. The resulting moderator tank’s outer surface resembles a hemi-sphere with radius 22 cm. The variation in the cost value with search iteration number and the resulting optimal shape are presented in Figs. 4.13-4.17.
Figure 4.13. Improvement in cost value by Min-max.2 in 97 search iterations.

The performance comparison of the optimal shape with the actual D$_2$O tank currently in operation at the PSBR shows that the optimal shape produces a larger partial current output at the exiting surface of the Bi disk and at the beam exit hole. These comparisons are shown in Figs. 4.18 and 4.19, respectively.

All optimization results presented so far illustrate that the moderator tank has almost hemi-spherical optimal shape, and therefore the existing drum-shaped D$_2$O moderator tank is over-designed. Furthermore, since the PSBR core is over-moderated due to the presence of ZrH in the core (in the form of fuel), the neutrons leaking out of the core into the D$_2$O tank are mostly well thermalized. Hence it is concluded that the principal function of the D$_2$O moderator material is to reflect thermal neutrons back into the core rather than moderating fission neutrons that escape the core. This is evident because during the optimization process from the fact that the moderator material clusters around the beam tube base (re-entry hole) As a consequence of this observation, the extent of the search problem domain can be reduced by restricting the D$_2$O-H$_2$O replacement to cells within a region around the beam tube base.
Figure 4.14. Optimal D$_2$O tank shape with Min-max-2.

Figure 4.15. Top view of the optimal D$_2$O tank shape with Min-max-2; not showing H$_2$O.
Figure 4.16. Back view of the optimal D$_2$O tank shape with Min-max-2; not showing H$_2$O.

Figure 4.17. Front view of the optimal D$_2$O tank shape with Min-max-2; not showing H$_2$O.
Figure 4.18. Comparison of outgoing current at the exit hole for the real and optimal D$_2$O tank shape with Min-max-2.

Figure 4.19. Comparison of outgoing current after Bi for the real and optimum D$_2$O tank shape with Min-max-2.
4.2 Performance Evaluation of the Parallel Execution of MOZAIK

In order to test the parallel performance of the code package, the same optimization problem described in the previous sections was executed by the serial and parallel versions of MOZAIK with Min-max-2. The results for this study are shown in Figs. 4.20 - 4.25.

The result shown in Fig. 4.20 indicates that the total execution time of serial MOZAIK increases with the number of optimization iterations. The Min-max replacement operation is responsible for this behavior since the number of states increases almost in proportion to the number of new cells that can be added to the search domain in every iteration. Figure 4.21 presents the average computation time per state for serial MOZAIK execution. In this execution, a TORT scalar flux initial guess file was generated using the best state data from the previous iteration to reduce TORT’s computation time for each state. Therefore, initially the average computation time for a state is long, but after a few iterations, it approaches its minimum value because the states of the two consecutive iterations become sufficiently similar, i.e. the difference between the material distributions is not significant, to yield rapid convergence. Then, the average computation time increases slowly until the number of states becomes sufficiently large per optimization iteration. After this point, the total execution time fluctuates around an approximately constant value.

The results shown in Figs. 4.22 and 4.23 illustrate the same effects for the parallel execution of MOZAIK. For this case, time was measured on the master process as the wall-clock time of each optimization iteration. Since the number of states increases during the optimization procedure, the load of each processor increases and therefore the wall-clock time exhibits a generally increasing trend. On the other hand, the average wall-clock time per optimization iteration is initially long, then it decreases to an almost constant value as the number of states increases in the optimization iteration.

Parallel MOZAIK execution used 32 identical processors for these calculations. In addition, the self scheduling strategy described in Section 2.6.2 was enabled to adjust the processors’ load during the optimization. The variation of the parallel
Figure 4.20. Variation of the total (i.e. cumulative) computation time for the optimization calculation with serial execution of MOZAIK

Figure 4.21. Average computation time per state for the optimization calculation with serial execution of MOZAIK
Figure 4.22. Variation of the total computation time for the optimization calculation with parallel execution of MOZAIK

Figure 4.23. Average computation time per state for the optimization calculation with parallel execution of MOZAIK
speedup and total processor idleness from one iteration to another are given in Figs. 4.24 and 4.25, respectively. Initially, the speedup is rather low, about a factor of 3 on 32 processors, because the number of states in the first few optimization iterations is smaller than the number of processors and this causes uneven computational load distribution among the participating processors. After a few iterations, the speedup increases to 28 as the number of states, hence the potential for substantial concurrency, increases.

MOZAIK’s parallel efficiency approaches to 85% at 40 iterations. This value might increase and approach the theoretical efficiency of 100% when the number of states increases in further optimization iterations since in such case the processors’ idle time does not contribute significantly to the wall-clock time.

**Figure 4.24.** Speedup variation during the optimization procedure (32 processors)
4.3 Summary of Testing MOZAIK

The preliminary performance testing of MOZAIK presented in this chapter shows that the developed optimization sequence is feasible and ready for actual optimization calculations. The following results and general observations were obtained.

1. Model problem:

   (a) The preliminary results of the model configuration show that the source spatial distribution is not significant. Therefore, a uniform spatial distribution of the neutron boundary source on the core interface was used for this preliminary optimization calculation. However, the material distribution in the search problem domain affects the number of neutrons passing from the core to the search problem domain since the reflection and the absorption properties near the core interface change depending upon the D₂O-H₂O-air material distribution. Therefore, a comprehensive analysis is necessary for the boundary source generation.
2. Physics module:

(a) Shorter computation time required by TORT, in contrast to MCNP5, encourages usage of the former code in the optimization algorithm.

(b) MCNP without biasing is not well suited for the optimization search in this problem.

3. Optimizer Module:

(a) In preliminary testing of optimization, by means of the cluster replacement strategy, the Min-max optimizer runs on a coarser mesh than the fine mesh employed by the physics module. This might produce a shape that cannot be fabricated but has the favorable effect of reducing MOZAIK’s total execution.

4. Interface Blocks:

(a) Interface blocks between the optimizer and physics modules are operational.

5. Multi-processor support:

(a) The parallel version of MOZAIK is operational, and yields speedup at reasonable parallel efficiency on up to 32 processors in a homogeneous multiprocessor environment.

(b) The parallel performance of parallel MOZAIK with the self-scheduling strategy improves with increasing number of states.

(c) Self-scheduling is essential to provide better load balancing among the participating processors.
Chapter 5

Optimization Calculations with MOZAIK

For the PSBR design calculations, three optimization model problems were designed to improve the performance of the beam port facility: (1) determining the optimal shape of the D$_2$O moderator tank for the PSBR with the existing tangential beam tube configuration, (2) determining the optimal shape of the moderator tank for the PSBR with a tangential beam configuration in which the beam tube adjacent to the core interface, and (3) determining the optimal shape of the moderator tank for the PSBR with a radial beam tube configuration. Optimal D$_2$O moderator tank shapes were determined by MOZAIK for each of the three model problems.

In this chapter, the results of the optimization calculations for the three model configurations then performance evaluations of the three obtained optimal beam tube configurations, and performance analysis of MOZAIK for these optimization calculations are analyzed.
5.1 First Mission: Determining the Optimal Shape of the D$_2$O Moderator Tank for the Existing Beam Port of PSBR

The optimization problem, determining the optimal shape of the D$_2$O moderator tank for the PSBR with the existing beam tube, was assigned to MOZAIK as its first mission.

First a neutronic model including the existing beam tube configuration was constructed without the existing D$_2$O. In addition, a computational study was designed to determine the ideal mesh size for this neutronic model that yield to obtain reasonably accurate results at a reasonable computational cost, i.e. time and memory requirements.

After fixing the neutronic model with its auxiliary components, the optimization model was developed and MOZAIK’s modules were prepared for this optimization calculation. Preliminary results were obtained by MOZAIK including the physics module with constant boundary source approach introduced in Section 3.5.2, and a preliminary verification study was performed to evaluate MOZAIK’s results and performance. The results of the verification study showed that a new source generation strategy is inevitable. This is a consequence of the changing the material distribution in the search problem domain during the optimization calculation, that significantly affects the neutron boundary source applied at the core-tank interface. Therefore, a new source module was developed to generate a material-dependent boundary source distribution on-the-fly during the optimization calculation. Furthermore, in order to speed up the optimization calculation, a study was done to determine the extent, i.e. the dimensions, of the search problem domain. Finally, the optimal D$_2$O tank shape for this problem was determined by MOZAIK with the two optimizers, namely Min-max and Genetic Algorithm.

In this section, all these developmental stages and the results computed by MOZAIK are described.
5.1.1 Neutronic Model

For this optimization problem, the neutronic model presented in Section 4.1.1 was modified by fixing the beam tube location as the same location at the existing tangential beam tube BT#4 in the PSBR location:

1. The geometry of the beam tube re-entry hole and its location were fixed to coincide with the existing beam tube configuration.

2. All aluminum components, e.g. moderator drum, aluminum around the beam re-entry gap, etc, were removed from the Tank Model. In addition the graphite block behind the D$_2$O moderator tank was also removed from the model.

3. The Bi disk was placed at the junction between the beam re-entry gap and beam tube.

4. During the optimization, two options were exercised for the boundary source distributions:

   (a) Fixed boundary source: The boundary source is held fixed while the material distribution in the search problem domain changes by state in the optimization procedure. The boundary source (spatial, angular and energy distribution) was computed using the MCNP Core Model with the material configuration at the initial state from which the optimization sequence is launched.

   (b) Material dependent boundary source: The boundary source is recalculated by MCNP as the material distribution changes in search problem domain during the optimization sequence. A new source module was designed to perform this calculation for each state separately. The description of this module and the new source generation strategy is presented in Section 5.1.6

The neutron beam intensity at the beam tube exit end for the existing beam tube configuration of PSBR was also computed by TORT and the streaming operator, and this result was denoted as the reference result used to evaluate the
improvement achieved by the optimization calculations. In this reference calculation, TORT used the same settings that were used for the TORT calculation in the optimization sequence such as mesh size, angular quadrature, inner/outer iteration numbers, etc., similar to the settings for the TORT Tank Model introduced in Section 3.2.2. Hence, further sources of discrepancy between the exercised models were eliminated in the optimization results except for the effects of the source approximation.

5.1.1.1 Mesh Size Selection

A study was designed to determine an ideal computational mesh size for the neutronic model. The term “ideal” implies a the mesh structure was selected from a set of mesh structures (from coarsest to finest) that produces reasonably accurate results in adequately short execution times.

Initially a set of Cartesian mesh structures were generated for the neutronic model. Then, TORT input files for the neutronic model given in this section were generated using these mesh structures.

For the next step, TORT and streaming calculations were performed to compute the neutron partial current output at the beam tube exit end for the neutronic model with different mesh structures. The model with the finest mesh was denoted as the reference model for this study and its result, thermal neutron output at the beam tube exit end, was used to evaluate the performance of the coarser-mesh models. As a result of this calculation, a judgement can be made to determine the most efficient mesh structure, i.e. highest accuracy for a given execution time, for the full optimization calculations.

For this purpose, the set of Cartesian mesh structures given in Table 5.1 were generated for the neutronic model. In order to improve the streaming results, the mesh in the beam tube center region, whose radius is almost 1 cm, was refined for all mesh structures. Then, TORT input files using these mesh structures were generated. Finally, TORT and streaming calculations were performed for all listed models, and the corresponding computed beam intensities at the beam tube exit hole its results were compared to the value obtained with the reference model, i.e. 83,720 cells.
Table 5.1. Several Cartesian meshes employed in determining an efficient mesh structure for the physics module

<table>
<thead>
<tr>
<th>Total Cell</th>
<th>Mesh numbers</th>
<th>Mesh Sizes</th>
<th>Fine Mesh at the BT center</th>
</tr>
</thead>
<tbody>
<tr>
<td>7920</td>
<td>18×20×22</td>
<td>3.84 × 4.00 × 3.84</td>
<td>4 × 4</td>
</tr>
<tr>
<td>9504</td>
<td>18×24×22</td>
<td>3.84 × 3.84 × 3.84</td>
<td>4 × 4</td>
</tr>
<tr>
<td>10764</td>
<td>18×26×23</td>
<td>3.84 × 3.50 × 3.50</td>
<td>4 × 4</td>
</tr>
<tr>
<td>13800</td>
<td>20×30×23</td>
<td>3.36 × 3.44 × 3.50</td>
<td>8 × 6</td>
</tr>
<tr>
<td>16200</td>
<td>20×30×27</td>
<td>3.36 × 3.44 × 3.35</td>
<td>8 × 6</td>
</tr>
<tr>
<td>20416</td>
<td>22×32×29</td>
<td>2.88 × 3.24 × 3.20</td>
<td>8 × 6</td>
</tr>
<tr>
<td>21344</td>
<td>23×32×29</td>
<td>2.88 × 3.24 × 3.20</td>
<td>8 × 6</td>
</tr>
<tr>
<td>24012</td>
<td>23×36×29</td>
<td>2.88 × 2.77 × 3.20</td>
<td>8 × 6</td>
</tr>
<tr>
<td>27324</td>
<td>23×36×33</td>
<td>2.88 × 2.77 × 2.81</td>
<td>8 × 6</td>
</tr>
<tr>
<td>44688</td>
<td>28×42×38</td>
<td>2.30 × 2.50 × 2.45</td>
<td>10 × 8</td>
</tr>
<tr>
<td>83720</td>
<td>35×52×46</td>
<td>1.92 × 2.02 × 1.98</td>
<td>14 × 12</td>
</tr>
</tbody>
</table>

The results of this study are presented in Figs. 5.1 and 5.2. TORT computational time increases in proportional to the number of cells. In contrast, the percent difference between the neutron beam intensity computed on the finest-mesh (reference) model and on the results of various coarse-mesh models is less than 12%. This result shows that performing the optimization calculations using one of the coarse meshes attempted is suitable because it produces as reasonably accurate results as the finest model without consuming too much time.

5.1.2 Model Development for the Optimization Calculations

For the optimization problem, the model configuration was defined by:

1. Removing the D$_2$O moderator from the system and filling the problem domain with water.

2. Placing the beam tube at the same location as the existing tangential beam tube configuration of PSBR.

3. Placing the Bi gamma shielding disk at the junction between the beam re-entry gap and the beam tube.
4. In order to speed up the optimization calculations, MOZAIK’s dual mesh option was enabled: (1) $12 \times 22 \times 22$ Cartesian grid for the optimizer, and (2) $18 \times 28 \times 24$ Cartesian grid for the physics calculation.

![CPU Time variation in terms of number of computational cells](image)

**Figure 5.1.** CPU Time variation in terms of number of computational cells

### 5.1.3 Definition of the Cost Function

When performing the cost calculation in MOZAIK for this model problem, the thermal energy threshold cutoff was set to 0.625 eV and the contribution of neutrons having higher energy to the cost value was diminished, while the contribution of neutrons having lower energy than 0.625 eV was magnified. The cost function given in Eq. 4.1 was used with the above energy cutoff value for both optimization strategies, i.e. Min-max and GA. In this way, the cost of each state is computed and compared to the other states’ cost by the optimizer to find the optimal state which has the minimum cost value.
Figure 5.2. Relative difference between the exit hole neutron partial current computed on a coarse mesh model compared to the value computed with the reference model’s fine mesh.

5.1.4 Physics Code Preparation

The same TORT pre-processor and post-processor that are introduced in Chapter 4 for the validation calculations of MOZAIK were prepared for this optimization calculation. The new source module was also assembled to the physics module in the optimization sequence.

The *gentinp-3* code was used to generate the TORT input file for a given material distribution (state) and the TORT input template. The following parameters were entered in the TORT input template:

1. Solution method, \((\theta\text{-weighted with } \theta=0.9)\).
2. Angular quadrature (Upward-biased-100).
3. Maximum number of outer and inner iterations, 300 and 4 respectively.
4. Maximum number of directions in the angular quadrature set (100).
5. Point-wise flux convergence criteria (1.e-3).
6. Source iteration convergence criteria (1.e-3).

7. All cross-section control parameters consistent with the cross-section library data.

5.1.5 Limiting the Size of the Search Problem Domain

In order to determine the limits of the extent of the search problem domain, a simple decomposition methodology was defined to determine the contribution of the neutrons originating at various locations within the problem domain to the beam tube exit flux by placing a thin highly black absorber material around the beam tube in the moderator tank. For this purpose, a simplified form of the existing tangential beam tube of the PSBR was modeled by TORT.

In the model configuration, the $\text{D}_2\text{O}$ moderator material was replaced with various materials, and the effect of these changes on the beam tube exit flux was observed and analyzed. In this way, sufficient information was acquired for restricting the extent of the search problem domain for the optimization problem thus reducing the computational effort of the search algorithm.

A similar computational experiment was performed for the radial beam tube configuration which is planned for the new beam tube arrangement of the PSBR to determine the contribution of neutrons originating at various locations to the beam tube exit flux and to evaluate the effect of the location of the Bi gamma shielding disk on the exit flux.

5.1.5.1 Decomposition Methodology

1. Tangential Beam Tube Configuration

In this case the origin of the beam exit flux was expressed in only two components: forward scattering and scattering components,

$$\psi_e(r, E, \mu_0) = \psi^{\text{Base}}_{\text{Bi}}(r, E, \mu_0) + \psi^{\text{Side}}_{\text{Bi}}(r, E, \mu_0)$$  \hspace{1cm} (5.1)

The black absorbers were placed and arranged around the beam tube re-entry gap to compute flux components easily.
In these calculations, first a full-scattering source was computed for the model configuration without the black absorbers. This was done by running TORT till convergence, then folding the converged cell-averaged scalar flux with the scattering matrix to obtain the cell-wise scattering source by energy group. Then the black absorbers were placed in selected locations in the model and only one inner iteration was performed by TORT with this scattering source acting as a distributed source to compute each component of the angular flux exiting the beam tube exit end. In this methodology, the sum of the components overestimates the total exiting flux since both components include the same scattering source in the re-entry gap. Hence, a third component was subtracted to eliminate this double-counting,

\[ \psi_e(r, E, \mu_0) = \psi_{B_i}^{Base}(r, E, \mu_0) + \psi_{B_i}^{Side}(r, E, \mu_0) - \psi_{B_i}^{Base,Side}(r, E, \mu_0) \]  

A schematic of the model configuration of the tangential beam tube is shown in Fig. 5.3. It includes only the D\(_2\)O drum enclosed in an aluminum cylindrical chamber, the re-entry gap of the beam tube, the Bi disk and two different black absorber locations.

![Figure 5.3. Sketch of the model problem configuration for the tangential beam tube showing locations of the black absorbers](image)

The following configurations were prepared to determine the components of
the thermal neutrons exiting angular flux:

(a) By placing a thin highly absorbing disk B, behind the beam re-entry gap in the moderator drum, $\psi_{\text{Base}}^{\text{Bi}}(r, E, \mu_0)$ can be blocked (see Fig. 5.3), hence the scattering component, $\psi_{\text{Side}}^{\text{Bi}}(r, E, \mu_0)$, dominates the exiting neutron beam.

(b) By covering the beam tube side-walls with a thin highly absorbing material A, $\psi_{\text{Side}}^{\text{Bi}}(r, E, \mu_0)$ can be blocked (see Fig. 5.3), hence $\psi_{\text{Base}}^{\text{Bi}}(r, E, \mu_0)$ dominates the neutron output.

(c) By placing absorbers A and B at the same time, only neutrons in the re-entry gap contribute to the exit flux, hence the reduction term, $\psi_{\text{Base,Side}}^{\text{Bi}}(r, E, \mu_0)$, is computed.

As a result, by comparing the magnitude of the neutron beam exiting the beam tube with the various black absorber arrangements for the same moderator material, the effects of various moderator materials on the contributions of various locations in the search problem domain to $\psi_e$ were quantified and compared. Using the results of this comparison the search problem domain space can be reduced in size thereby speeding up the optimization calculation.

2. Radial Beam Tube Configuration

The origination point for thermal neutrons comprising the beam tube’s exit angular flux was expressed in three components: uncollided, forward scattering and scattering components,

$$\psi_e(r, E, \mu_0) = \psi_{\text{unc}}(r, E, \mu_0) + \psi_{\text{Base}}^{\text{Bi}}(r, E, \mu_0) + \psi_{\text{Side}}^{\text{Bi}}(r, E, \mu_0)$$ (5.3)

The uncollided component, $\psi_{\text{unc}}(r, E, \mu_0)$, is the flux of almost mono-directional neutrons moving parallel to the beam tube axis, having has their last scattering collision in the moderator materials (e.g. H$_2$O or D$_2$O) and passing through the Bi disk without any collision. The forward scattering component, $\psi_{\text{Base}}^{\text{Bi}}(r, E, \mu_0)$, is defined as the neutrons which are moving parallel to the beam tube axis, coming to the exit surface of Bi from the beam
tube base and having the last scattering collision in the Bi disk. The third component, $\psi_{\text{Side}}^{\text{Bi}}(r, E, \mu_0)$, is defined as the neutrons which are moving parallel to the beam tube axis, coming to the exit surface of Bi from everywhere except the beam tube base and having the last scattering collision in the Bi disk. Since dry air inside the beam re-entry gap has low collision probability, it does not significantly affect the uncollided and forward scattering component while neutrons are streaming from the beam tube base to the incoming surface of the Bi disk.

In order to compute these components separately, a set of black absorber configurations were prepared for the model configuration of the radial beam tube.

A schematic of the model configuration of the radial beam tube is shown in Fig. 5.4. It includes only the D$_2$O drum enclosed in an aluminum cylindrical chamber, the re-entry gap of the beam tube, a Bi disk used for gamma shielding before the beam tube entrance and three different black absorber locations.

The following configurations were prepared to compute each of the three components of the neutron output:

(a) By placing absorber A and C at the source interface, $\psi_{\text{Base}}^{\text{Bi}}(r, E, \mu_0)$ and $\psi_{\text{unc}}(r, E, \mu_0)$ were blocked (see Fig. 5.4), hence the scattering component, $\psi_{\text{Side}}^{\text{Bi}}(r, E, \mu_0)$, dominates the exiting neutron beam.

(b) By placing absorber A and B at the source interface, $\psi_{\text{Base}}^{\text{Bi}}(r, E, \mu_0)$ and $\psi_{\text{Side}}^{\text{Bi}}(r, E, \mu_0)$ were blocked (see Fig. 5.4), hence the uncollided component, $\psi_{\text{unc}}(r, E, \mu_0)$, dominates the exiting neutron beam.

(c) By placing absorber B and C at the source interface, $\psi_{\text{Base}}^{\text{Bi}}(r, E, \mu_0)$ and $\psi_{\text{unc}}(r, E, \mu_0)$ were blocked (see Fig. 5.4), hence the forward scattering component, $\psi_{\text{Side}}^{\text{Bi}}(r, E, \mu_0)$, dominates the exiting neutron beam.

In these configurations, each component was computed by directly blocking the source neutrons which generate the other two components. By evaluating this information, the search space can be reduced for the optimization
problem. Furthermore, by changing the location of the Bi disk along the beam tube, its optimal location can be determined easily.

### 5.1.5.2 Numerical Results

In order to test the methodology, one of the existing tangential beam tubes of the PSBR was modeled with a D$_2$O moderator at the beam tube entry; this is designated as the reference configuration, and various moderator materials such as H$_2$O, and graphite were attempted in the tank. Moreover, as an additional study, the same calculations were performed on the model with the moderator tank filled with dry air to determine the reflection effect inside the moderator tank. Then, all calculations were repeated to compute each component of the angular flux exiting the beam tube separately by placing black absorber(s) at the selected locations described above to absorb the other components.

An analogous set of computational experiments were performed on the model for the radial beam tube.

All results are given in Tables 5.2-5.3. In the tables, $\psi_{exit}$ denotes the exit flux.
computed by TORT plus the streaming operators and $\psi_e$ denotes the exit flux calculated by summing the components of the flux. The results show that the neutron output of the radial configuration with the same moderator tank, beam tube, and interface source properties is almost 2.5 times larger than the neutron output of the existing tangential beam tube configuration. The best moderator material is D$_2$O for both model configurations.

The results illustrate that the contribution of the forward scattering component of the exit flux is dominant for both radial and tangential configurations for various moderator materials. Specifically, more than 80% of the thermal beam neutrons originate at the beam tube base and make their last scattering collision in the Bi disk before streaming down the beam tube to the exit hole. These results explain the tendency of the optimization algorithm in preliminary studies to concentrate much of the D$_2$O-H$_2$O replacement in the region around the beam tube base. Armed with the results of this study, the extent of the search problem domain can be reduced to only that region thereby significantly reducing the number of computational cells whose material content must be determined to improve the beam quality.

<table>
<thead>
<tr>
<th>Moderator Material</th>
<th>Flux per energy ($n/cm^2 - eV$)</th>
<th>Fractional contribution (%)</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D$_2$O</td>
<td>5.506e-09</td>
<td>5.404e-09</td>
<td>100 × (\psi_{exit} - \psi_e)</td>
</tr>
<tr>
<td>H$_2$O</td>
<td>1.913e-10</td>
<td>1.846e-10</td>
<td>1.85</td>
</tr>
<tr>
<td>graphite</td>
<td>4.011e-09</td>
<td>3.880e-09</td>
<td>3.02</td>
</tr>
<tr>
<td>air(dry)</td>
<td>3.346e-09</td>
<td>3.305e-09</td>
<td>1.22</td>
</tr>
</tbody>
</table>

Table 5.2. Angular flux and its components in the thermal groups computed by TORT + streaming operator at the beam tube exit hole for the tangential beam tube for various moderator materials
Table 5.3. Angular flux and its components in the thermal groups computed by TORT + streaming operator at the beam tube exit hole for the radial beam tube for various moderator materials

<table>
<thead>
<tr>
<th>Moderator Material</th>
<th>Flux per energy $(n/cm^2 - eV)$</th>
<th>Fractional contribution (%)</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\psi_{\text{exit}}$</td>
<td>$\psi_e$</td>
<td>$\psi_{\text{Base}}$</td>
</tr>
<tr>
<td>D$_2$O</td>
<td>1.335e-08</td>
<td>1.312e-08</td>
<td>90.99</td>
</tr>
<tr>
<td>H$_2$O</td>
<td>1.041e-08</td>
<td>1.022e-08</td>
<td>92.29</td>
</tr>
<tr>
<td>graphite</td>
<td>1.259e-08</td>
<td>1.244e-08</td>
<td>91.72</td>
</tr>
<tr>
<td>air(dry)</td>
<td>7.185e-09</td>
<td>7.157e-09</td>
<td>85.34</td>
</tr>
</tbody>
</table>

5.1.6 New Source Module: Material Dependent Source Generation

The first set of the optimization calculations were conducted with the fixed boundary source approach. After completing these optimization calculations, validation exercise was performed to test and verify the optimization results. For this purpose, the best state data (material distribution) of each search iteration was deployed in MCNP core models, and real boundary sources were computed consistently by the source module for these different states.

Similarly, TORT models were designed using these states. By executing TORT for these models with the generated real boundary sources, and applying the streaming operator, the outgoing partial current of thermal neutrons for each state was computed at the beam tube exit end. The comparison of the outgoing partial currents computed with these two approaches is shown in Fig. 5.5. The results illustrate that the error increases with the number of iterations. Because the new tank shape in each iteration contains much more D$_2$O cells than the initial state, the fixed boundary source approach increasingly inaccurately represents the boundary source for this calculation. In Fig. 5.5, the “Reference Output” represents the outgoing partial current computed by TORT and streaming operator at the beam tube exit end for the current beam port configuration and the D$_2$O moderator tank.

As a result of this, a study was performed to test the material distribution effects on the neutron boundary source generated by the source module from the
Figure 5.5. Comparison of the thermal neutron beam intensities outputs computed by TORT and the streaming operators for both the fixed source approach and the real boundary source.

MCNP full model’s SSW data file. By replacing the D$_2$O moderator material with alternative relevant materials in the MCNP full model, a few MCNP models were designed to investigate the effect of the material distribution on the boundary source distribution. The new models are:

1. Original model (MCNP Core Model described in Chapter 3), tank with D$_2$O
2. Original model, tank with air
3. Original model, tank with graphite
4. Original model, tank with water
5. Original model, empty tank (vacuum)

Then, MCNP simulations were performed for these models to generate SSW neutron data files for the boundary source generation. The source data were profiled
in terms of their spectral distribution, and the magnitude of the source, by the source module.

First, the magnitude of the source neutrons per fission event, i.e. the number of neutrons passing from core side to the tank side at the core-tank interface, was calculated to quantify the effect of the moderator material effects on the number of source neutrons. The results reported in Table 5.4 indicate that the D$_2$O and graphite materials increase the source neutron magnitude because they are more diffusive than the water and they are better reflectors than air. Although water reflects neutrons back to the core, the number of neutrons is not as high as when the D$_2$O is used because of the higher absorption cross-section in water. For the models with air and vacuum, the source neutron magnitude is too low due to the very low rate of reflection.

<table>
<thead>
<tr>
<th>MATERIAL</th>
<th>neutrons per fission event</th>
</tr>
</thead>
<tbody>
<tr>
<td>D$_2$O</td>
<td>0.13812</td>
</tr>
<tr>
<td>graphite</td>
<td>0.13324</td>
</tr>
<tr>
<td>water</td>
<td>0.11925</td>
</tr>
<tr>
<td>air</td>
<td>0.07329</td>
</tr>
<tr>
<td>void</td>
<td>0.07211</td>
</tr>
</tbody>
</table>

In the same way, neutron spectra computed by the source module for these different models were compared. Source neutron spectra in terms of 1 fission event for these models are shown in Fig. 5.6. Although the shapes of the spectra are roughly the same, the magnitudes are different as described above. This figure shows that D$_2$O is the best moderator material for this model problem in terms of the strength of the thermal neutron boundary source, and that air is the worst. In other words, air reduces the number of neutron emissions at the core-tank interface whereas the D$_2$O increases it.

As a consequence of this study, it is obvious that the material distribution in search problem domain affects the boundary source. Therefore a new source generation strategy is necessary to approximate the boundary source more precisely.
Figure 5.6. Comparison of source spectra (normalized to 1 fission event)

5.1.6.1 New Boundary Source Generation Strategy

The results presented in Chapter 3 demonstrate that it is impractical to generate material dependent boundary neutron source for each state by performing MCNP full model calculations during the optimization sequence. This conclusion follows from the fact that the MCNP full model calculations are computationally expensive and their execution times are not comparable to the time required to execute the physics module for of a state. Therefore, a computationally efficient strategy is necessary for generating the material dependent source.

One possible strategy is to express the source parameters in terms of the material distribution by performing artificial neural network calculation [52]. An artificial neural network (ANN), often just called a "neural network" (NN), is a mathematical model or computational model based on biological neural networks. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal informa-
tion that flows through the network during the learning phase. In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. One of the toolboxes for neural network applications is SNNS, Stuttgart Neural Network Simulator [53]. It is a simulator for neural networks developed at the Institute for Parallel and Distributed High Performance Systems, at the University of Stuttgart in 1989. The goal of the project is to create an efficient and flexible simulation environment for research on, and application of, neural networks.

The strategy for the new source module is as follows:

1. A set of states (material distributions) are randomly generated that includes \( \text{H}_2\text{O} \) cells, \( \text{D}_2\text{O} \) cells, and fixed air cells which represents the beam re-entry gap.

2. The boundary source for each state is computed by the previously described source module based on the MCNP Core Model. Then the spatial and energy distributions of the boundary source neutrons are separately obtained at the source interface.

3. The problem domain is divided into a number of regions, and material volume fractions (\( \text{H}_2\text{O} \) and \( \text{D}_2\text{O} \)) are calculated in each region.

4. A function/relation is obtained between the material volume fractions in each region and the neutron source spectrum by using SNNS. For this purpose, the material volume fractions in each region were prepared as input patterns for SNNS, and the 26 group source spectra were prepared as output pattern for SNNS. By selecting appropriate learning functions and a suitable set of parameters, a function was obtained that calculates the source spectrum for the given regional material volume fractions.

The verification of the accuracy of this approach in representing the boundary source in a computationally efficient way is presented in Fig. 5.7. For the verification calculations, a number of material distributions (states) were randomly generated and source spectra were computed by MCNP5 for the MCNP Core Models with these states, and by the source spectrum function
obtained by SNNS calculations. The percent difference between both calculations are shown in Fig. 5.7 for these randomly generated states. The results indicate that the most of the states have a difference less than 1.5% and the maximum difference is almost 2.5%. In other words, newly introduced source spectrum function improves the neutron boundary source.

5. The core-moderator interface is divided into a number of regions, and material surface area fractions (H$_2$O and D$_2$O) are calculated in each region,

6. A function/relation is obtained between the material area fractions in each region and the neutron spatial distribution by using the SNNS. For this purpose, the material area fractions in each region were prepared as input patterns for SNNS, and the source spatial distribution was prepared as its output pattern. By selecting appropriate learning functions and its appropriate parameters, a function was obtained that calculates the source spatial distribution for a given regional material area fractions.

The verification of the result of this function is shown in Fig. 5.8. For the verification calculations, a number of material distributions were randomly generated and source spatial distributions were computed both by the previous source module based on the MCNP Core Models, and by the source spatial distribution function obtained by SNNS calculations. The percent difference between both calculation strategies is given in Fig. 5.8. The results indicate that the most of the states have a difference less than 12.0% and the maximum difference is almost 17.0%. It can be useful for the calculations in which the beam tube is not closer to the source interface.

7. Multiplying the spectral distribution function by the spatial distribution function generates an approximate source distribution at the source interface in terms of the material distribution in the problem domain in terms of the regional material fractions.

Using the new source module, the validation study for the optimization calculation given in the beginning of this section was repeated. The same exiting thermal neutron beam intensity values were computed by TORT using the boundary source generated by the new source module, and the streaming operator at the beam
Figure 5.7. Difference between the spectra computed by the source spectrum function obtained by SNNS and the previous source module with the MCNP Core Model

Figure 5.8. Difference between the source spatial distributions computed by the source spatial distribution function obtained by SNNS and the previous source module with the MCNP Core Model
tube exit end. The comparison of the results for the real source and the new source module are presented in Fig. 5.9. In Fig. 5.9, the “Reference Output” represents the outgoing partial current computed by TORT and streaming operator at the beam tube exit end for the current beam port configuration and the D$_2$O moderator tank. Although the results show that there is still a difference between the results with source module and the results with the real source, this difference is of reasonably acceptable magnitude. As a result, the source should be re-generated for each state and the new source module thus facilitating the optimization calculation and improves the accuracy of MOZAIK’s results.

![Comparison of outgoing partial currents](image)

**Figure 5.9.** Comparison of the thermal neutron partial current at the exit end of the beam tube computed by TORT and the streaming operator using the SNNS-based source approach and the real source

### 5.1.7 Optimization Calculations

The objective of the optimization calculation was defined as determining a new D$_2$O moderator tank shape for the PSBR with the existing beam tube configuration that maximizes the thermal neutron beam intensity at the beam tube exit end. The TORT code was used as the neutron transport solver in the physics module
and it was executed on a $18 \times 24 \times 22$ Cartesian grid with an Upward-100 biased angular quadrature set, 26 energy groups, and $P_0$ isotropic scattering cross-section order. Two optimization algorithms, Min-max and Genetic Algorithms (GA), were used in the optimizer module to seek an optimal moderator tank shape for the given model configuration.

5.1.7.1 MOZAIK with Min-max

The optimization code with Min-max starts the calculation from an initial state at which the entire search problem domain was filled with only $H_2O$ cells. To generate the new states during the search, the material content of one of the $H_2O$ cells is replaced with $D_2O$. During the optimization procedure, Min-max takes the best state of the previous iteration and performs the replacement operation on either a single $H_2O$ cell or a group of $H_2O$ cells around the $D_2O$ cluster ($H_2O$ cells are replaced with $D_2O$) by using a breadth-first search strategy to obtain the new states. Finally, Min-max yields a $D_2O$ and $H_2O$ distribution over the cells in the search problem domain which maximizes the thermal neutron beam intensity. Here too, the replacement can be performed on a cell-by-cell or cluster-by-cluster basis and it is restricted to a region around the beam tube base to reduce the computational cost.

5.1.7.2 MOZAIK with GA

In contrast, GA starts with a number of initial states (128), each including randomly distributed $D_2O$ and $H_2O$ cells. During the optimization, GA performs its genetic operations, selection, cross-over and mutation, to generate new states, and finally obtains an optimal configuration. The GA parents-selection operator always selects the best state of the previous iteration as the first parent state and one random state from the states of the previous iteration as the second parent state. The cross-over operator performs a non-self avoiding random walk algorithm to determine the cross-over region between two states selected by the selection operator. After each cross-over operation, GA executes the mutation operator with a mutation probability which is designated as 0.7.
5.1.8 MOZAIK Results with the Two Optimizers

Results obtained by the two optimizers, Min-max and GA, and the reference value are shown in Figs. 5.10 and 5.11, respectively. The reference value, the outgoing thermal neutron flux at the exit hole for the existing tangential beam tube configuration, was computed by TORT as $4.41 \times 10^{-9} \, n/cm^2\cdot s$ (per fission event). Niederhaus reported the experimental value of the thermal neutron beam intensity (peak value) as $3.68 \times 10^{-9} \, n/cm^2\cdot s$ (per fission event) measured for the PSBR power at 1 MW level [31, 32]. The difference between the reference value computed by TORT and the measured value is due to the simplification of the beam port model such as removing the aluminum drum from around the moderator material and around the beam re-entry gap. Nevertheless this agreement is characterized as good agreement between the experimental and computational results.

![Comparison of cost values of two optimizers](image)

**Figure 5.10.** Cost values of the best states as computed by MOZAIK with the two optimizers

The optimally shaped moderator tanks determined by MOZAIK with the two optimizers, Min-max and GA, produce higher neutron outputs than the reference model. MOZAIK with Min-max determined the optimal moderator tank shape in 217 iterations a thermal neutron output beam intensity $4.83 \times 10^{-9} \, n/cm^2\cdot s$. 
whereas MOZAIK with GA determined its optimal moderator tank shape in 299 iterations with a slightly larger thermal neutron beam intensity $4.97 \times 10^{-9} \text{ n/cm}^2\cdot\text{s}$. These results indicate that the existing moderator tank design is not the best configuration to maximize the thermal neutron beam intensity even with the tangential configuration and a beam tube located almost 11 cm away from the core-tank interface. The new, optimal tank shapes determined by Min-max and GA have 9.5% and 12.8% higher beam intensity, respectively, compared to the existing configuration.

These results also indicate that Min-max found a best shape rather than an optimal shape. In other words, its search strategy is unable to find a global optimal shape for this model problem since its new states in any iteration grow only around the best state. In contrast, due to its diverse initial population and randomly generated new states, the GA-search covers the search space more comprehensively than Min-max. As a result, GA is more capable of escaping local extrema, and thus is better suited for finding an optimal moderator tank shape which maximizes the thermal neutron beam intensity. The other disadvantages of Min-max for
this model problem is that the computer time per search iteration increases with increasing search iterations. This is evident from the observed behavior of the two search algorithms as the number of new states increases shown in Figs. 5.12 and 5.13.

The effect of employing a flux initial guess strategy is also seen in Fig. 5.13. After a few iterations, the average TORT computation time per state decreases dramatically and fluctuates around 30 minutes per state. The gain in computation time is almost 1 order of magnitude in Min-max and a factor of 5 in GA.

![Variations of the number of states per iteration](image)

**Figure 5.12.** Number of states in each search iteration for MOZAIK with the two optimizers.

The optimal shapes found by MOZAIK with the two different optimizers are illustrated in Figs. 5.14-5.17. Although the optimization results show that the GA-search is more comprehensive than the Min-max, the optimal shape found by GA has more irregular D$_2$O structures than the shape found by Min-max. In other words, the fabrication of the GA’s optimal shape, or even a smoothed version of it, may turn out to be impractical even though it provides a higher thermal beam intensity.
Figure 5.13. Average computation time per state in each search iteration for MOZAIK with the two optimizers.

Figure 5.14. Front view (from core side) of the optimal shape found by MOZAIK using the Min-max optimization strategy.
Figure 5.15. Side view of the optimal shape found by MOZAIK using the Min-max optimization strategy.

Figure 5.16. Front view (from core side) of the optimal shape found by MOZAIK using the GA optimization strategy.
5.1.9 Smoothing Operation for the Optimal Shape

As mentioned in Chapter 2, originally a shape verifier module was developed as part of the optimization sequence, then it was removed from the optimization sequence, since the function of the verifier module (only post-processing operation on the results of the optimization procedure) is not integral with the optimization. Subsequently it was used to smoothen the produced optimal shape to make it more amenable to standard fabrication processes.

In the verifier module, the following strategy is used to smoothen the shape obtained by MOZAIK:

1. The center of mass coordinates of the converged optimal shape (in our case, the new D$_2$O tank shape) is calculated.

2. Distances between the cell-centers only of cells sitting at the shape’s outer boundary to the center of mass point are calculated. These values are used to split the shape into a few pieces to make data fitting easily.

3. A simple least-squares fit algorithm with the quadratic surface equation determines an approximate surface equation which represents the shape of each piece. Then, these surface equations are combined to represent the whole shape.
4. In this procedure, one of the constraints is to preserve the volume of the D2O tank and the surface area of the tank at the core face as closely as possible.

5. After obtaining the smoothed shape, the following calculations are performed to test this shape:

(a) MCNP Core Model is redesigned with this smoothed tank shape. Then, MCNP5 is executed with this model to obtain the necessary data for computing the neutron boundary source for the Tank Model.

(b) TORT Tank Model is redesigned with this smoothed tank shape by GGTM and gentinp-3.0, and TORT computes the neutron angular flux at the exit surface of the Bi disk using the neutron boundary source provided by the MCNP Core Model and the source module.

(c) The neutron beam intensity is computed by the streaming operator at the beam tube exit end. The result of the model including smoothed optimal tank shape is compared to the result of the optimal shape obtained by MOZAIK.

By means of this module, an approximate version of the optimal shape in a smooth structure is provided to the D2O tank designer and builder to make its fabrication possible.

This procedure was applied to smoothen the optimal tank shape determined by MOZAIK with Min-max for the existing beam port configuration of PSBR that is reported in the previous section. The 3D-views of the optimal shape are shown in Fig. 5.18. In the smoothing calculation, this shape was divided into two bodies: (1) a body whose center of mass point is near the beam tube base, (2) a body whose center of mass point is close to the middle of the beam re-entry gap. Two quadratic surfaces, each representing one of these bodies, were determined by the shape verifier module. The combination of these two surfaces constructs the final shape, the smoothed version of the optimal shape. These two surfaces and their combination are shown in Fig. 5.19. The first body is a part of hemi-sphere; although its radius is 30 cm, the radius of its intersection with the core interface is almost 27 cm. The second body was determined as part of a cylinder (smaller than a half-cylinder) around the beam tube re-entry gap, its radius is 23 cm and
its height is 20 cm. The combination of these two bodies resembles a bottle with a wide neck.

For the optimal shape determined by MOZAIK with Min-max, the neutron partial current in the thermal group was computed as $4.83 \times 10^{-9} \text{n/cm}^2\text{-s}$ at the beam tube exit end. The same quantity was computed for the smoothed shape as $5.84 \times 10^{-9} \text{n/cm}^2\text{-s}$ at the beam tube exit end. The 19% difference between the results of the two models (with/without smoothing) is due to the usage of the coarse mesh structure with TORT in the optimization calculations, the approximate boundary source computed by the source module and the approximations used in the shape smoothing procedure. In Chapter 2, it is stated that the difference in the TORT results obtained on the coarse mesh and fine mesh structures is almost 13%. Therefore, the approximations in this simple smoothing procedure yields a more amenable shape which produces the neutron beam intensity at the beam tube exit end with a computational error in the range of 6 – 7%. Furthermore, the results show that the PSBR’s beam tube #4 with this new tank design (smoothed optimal shape) produces higher thermal neutron beam intensity at the beam tube exit end.

Although the smoothing calculations were performed for the optimal shape determined by MOZAIK with genetic algorithms, neither a reasonable shape (produces neutron beam intensity with a small error) nor a regular shape (easily manufactured) could be obtained. This is due to the fact that the outer surface of the optimal tank shape is comprised of a lot of irregular D$_2$O structures, i.e. wiggly structures. For this reason, with this simple smoothing procedure it is impossible to obtain successful results.
5.2 New Beam Port Configurations

For the PSBR design calculations, two new beam tube configurations were investigated: (1) tangential beam tube configuration: the beam tube was moved to a position immediately adjacent to the core surface, and (2) radial beam tube configuration: the beam tube was rotated so its axis is normal to the core face. In both new configurations the dimensions of the beam tube were kept the same as in the existing design. In this section, these two configurations are described and their optimal shape designs obtained via MOZAIK are reported.

5.2.1 Tangential Beam Tube Adjacent to the PSBR’s Core Interface

The preliminary studies presented in Chapter 4 show that the thermal neutron beam intensity increases with moving the beam tube from its present location...
within the D$_2$O drum to a position closer to the PSBR core face. This conclusion was reached by performing manual optimization calculations using a set of pre-selected models obtained by varying the beam tube position within the existing D$_2$O tank model.

The goal of the optimization calculation described below is to determine an optimal tank shape for the configuration including a tangential beam tube expected to yield the maximal thermal neutron beam intensity at the beam tube exit end, i.e. with the beam tube placed as closed as possible to the core interface. The dimensions of the beam tube, the re-entry gap and, the Bi disk were fixed as those in the existing beam tube configuration. The beam tube location was set slightly away from the core interface in order to accommodate the core support structure.

For this calculation, Min-max without pruning was used in the optimizer module of MOZAIK because of its success in the optimization calculations reported in the previous section. Strategies similar to those described in the previous section were used to determine the mesh size, type and size of the angular quadrature, the search problem domain and the cost function. Specifically, the TORT code was used as the neutron transport solver in the physics module and it was executed on a $19 \times 28 \times 24$ Cartesian grid with an Upward-100 biased angular quadrature set, 26 energy groups, and $P_0$, i.e. isotropic, scattering cross-section data.

In the previous optimization calculations, a material dependent source module was specifically developed for the model employing the existing beam tube configuration with the re-entry gap buried deep into the D$_2$O tank. In contrast, in the new configuration considered in this section the beam tube’s re-entry gap introduces a vacuum near the core interface and this affects the spatial, angular and spectral distribution characteristics of the neutron boundary source at the core-tank interface. Therefore, a new material dependent source module was developed for this configuration by following the same source generation strategy described in the previous section.

The results of this optimization calculation are presented and discussed below.

5.2.1.1 Results of MOZAIK with Min-max Optimization

The results obtained by MOZAIK with Min-max in 138 optimization iterations, as well as those obtained by TORT for selected states during the optimization se-
quence using source data distributions, i.e. obtained via MCNP5 for the full Core Model are shown in Fig. 5.20. The reference value computed for the existing beam tube configuration is also presented in Fig. 5.20 for comparison purposes. The improvement in the thermal neutron beam intensity is almost 41% with the optimal beam tube design for this modified configuration. This beam tube arrangement with the optimal tank shape determined by this calculation shows better performance than the existing beam tube configuration with the optimal tank shape determined by MOZAIK with both optimizers that produced a gain in beam intensity of almost 28%. Figure 5.21 shows the amount of \( \text{D}_2\text{O} \) added by Min-max during the optimization procedure. For the optimal tank shape, the volume of \( \text{D}_2\text{O} \) is almost 1/3 of the current moderator tank’s volume.

The results shown in Fig. 5.20 comparing the beam intensity based on the actual source distribution versus based on the distribution computed with the source module, indicate that the source module developed for this configuration is slightly less accurate in representing the material dependency of the source distribution. In some cases, the difference in the computed beam intensities approaches 23%. This might be due to feeding insufficiently many input and output data values for SNNS to obtain the relation between them with high accuracy.

During the optimization procedure, in order to compute the thermal neutron beam intensity more accurately, a new source generation strategy was developed for this case. In this approach, the boundary source is directly generated every five optimization iterations by running the MCNP Core Model. It is evident that the adaptive source generation increases the computational load, and to some extent the total execution time. However, in this way the determined beam intensities can be computed more accurately since the source is computed directly using the actual state’s material distribution in the search problem domain, thus making the optimization result more reliable. MOZAIK with this source generation strategy performs the following steps:

1. Initially, the optimization calculation is started with an initial boundary source obtained by the source module using data provided by the MCNP Core Model for the initial configuration.

2. Typically, the difference between the material distributions of the best states
for two consecutive iterations is insignificant because of the Min-max replacement strategy. Therefore, during the optimization procedure a single approximate neutron boundary source is computed using the MCNP Core Model with the geometric configuration of the best state at the conclusion of iteration $I$, and is re-used for the next $I + IF$ search iterations. The term $IF$ is defined as the source generation frequency determined and set by the user (in our case, it was set to 5).

3. MOZAIK conducts the search sequence via Min-max as usual, then after $IF$ search iterations are completed, a new neutron source is generated using the best state’s material distribution in the search problem domain from the $I + IF$th iteration per step 2 above.

4. Steps 2 and 3 are repeated until MOZAIK determines an optimal shape.

One of the disadvantages of this strategy is that the parallel MCNP version does not permit the usage of the SSW feature that is necessary for the functionality of
Figure 5.21. Amount of D₂O added into search problem domain by Min-max during the optimization procedure.

The thermal neutron beam intensity obtained by MOZAIK with Min-max via the adaptive source generation strategy in 140 optimization iterations, and the beam intensity obtained by TORT for selected states using source data are shown in Fig. 5.22. The reference value computed for the existing beam tube configuration is also presented in Fig. 5.22 for comparison purposes. The results show that the adaptive source generation approach works well with MOZAIK and Min-max search technique, namely the computed beam intensities using adaptive and actual source distributions are in excellent agreement across the entire span of the search iterations. In contrast, since genetic operators produce two new individuals...
(states) randomly using individuals selected from the previous iteration, the source distribution is likely to change substantially from one search iteration to the next. Hence, the usage of this source generation approach is not suitable for MOZAIK optimization based on genetic algorithms.

**Figure 5.22.** Thermal neutron beam intensity of the best states at the exit hole that computed by MOZAIK with Min-max and adaptive source generation approach for the new tangential beam tube configuration, and comparison with values for selected states obtained using TORT with actual boundary source distribution.

The performance comparisons of MOZAIK with these two source generation approaches are given in Figs.5.23 through 5.25. In these figures, the adaptive source generation approach is labeled as SM-10.0 and the standard source module is labeled as SM-9.0. The results show that MCNP execution for SM-10.0 increases the computational load significantly every five optimization iterations, the value that IF was set to in our numerical experiments. However, to make the performance comparisons more fair, it is noted that the material dependent source generation step using SNNS requires substantial execution time because a number of random states were randomly generated then the MCNP Core Model was applied to their solution in order to train the neural network. This source generation time is not
included in the SM-9.0 plot of Fig. 5.23. The speed-up per search iteration approaches 30 for the parallel execution of MOZAIK on 32 processors using both source generation approaches, SM-9.0 and SM-10.0. The parallel performance of the code package improves, that is the speedup increases, with increasing number of states per each search iteration. The results also indicate that the adaptive source generation technique is feasible for this kind of calculation.

Figure 5.23. Comparison of the total execution time for MOZAIK with the adaptive (SM-10.0) and SNNS-based (SM-9.0) source generation strategies.

The optimal shape found by MOZAIK with Min-max for this alternative beam tube arrangement is depicted in Fig. 5.26. This shape was smoothed by the procedure summarized in the previous section and a smoothed optimal shape was obtained for the model with this beam tube configuration. The smoothed optimal shape shown in Fig. 5.27 consists of two parts; (1) hemi-spherical cap whose radius is 30 cm, and (2) a half cylinder around the beam re-entry hole whose radius is 17.2 cm and height is 26 cm. The resulting smoothed shape, shown in Figs. 5.27 and 5.28, is similar to the smoothed shape in the previous optimization calculation, see Fig. 5.19, except for the axis of the half cylindrical part. In this configuration,
Figure 5.24. Speed-up per search iteration of the parallel execution of MOZAIK on 32 processors with the adaptive (SM-10.0) and SNNS-based (SM-9.0) source generation strategies.

Figure 5.25. Processor idleness per search iteration for the parallel execution of MOZAIK with the adaptive (SM-10.0) and SNNS-based (SM-9.0) source generation strategies.
there is a significant off-set between the beam tube’s axis and the half cylindrical part’s axis. Because the neutrons’ spatial distribution is not symmetric vertically due to the control rods driving from the top of the core to its bottom, and this immediately affects the exiting beam intensity due to the relative position of the beam tube, i.e. too close, to the core interface.

For the optimal shape determined by MOZAIK with Min-max, Fig. 5.26, the neutron partial current in the thermal group at the exit hole was computed as $6.12 \times 10^{-9} \text{n/cm}^2\text{-s}$. The same quantity was computed for the smoothed shape, Fig. 5.28, as $7.13 \times 10^{-9} \text{n/cm}^2\text{-s}$. The 14% difference between the results of the two models (with/without smoothing) is due to the usage of a coarse mesh structure for TORT in the optimization calculations versus the fine mesh applied to the calculation with the smoothed shape, as well as the numerical errors incurred in the physics solution methods and algorithms.

**Figure 5.26.** 3D-views of the optimal D$_2$O tank shape determined by the optimization calculations of MOZAIK with Min-max(before smoothing)

**Figure 5.27.** 3D-views of the D$_2$O tank optimal shape smoothed by the shape verifier module (pink shows the smoothed optimal shape)
5.2.2 Radial Beam Tube

For the second new beam tube alternative configuration, a radial beam tube whose axis is normal to the core face was selected and a moderator tank shape was sought to maximize the thermal neutron beam intensity at the beam tube exit end. The dimensions of the beam tube, the re-entry gap and the Bi disk were fixed as in the existing beam tube configuration of the PSBR. Two different cases were investigated: (1) Radial beam tube configuration-I: Beam tube base is 11 cm far from the core face (direct rotation of the beam tube from the existing configuration, the rotation axis is at the beam tube base) (2) Radial beam tube configuration-II: Beam tube base is moved closer to the core face (distance is almost 3cm).

For these calculations, Min-max without pruning was used in the optimizer module of MOZAIK. Similar strategies described in the previous section were used to determine the mesh size, type and order of the angular quadrature, the search problem domain and the cost function. The TORT code was used as the neutron transport solver in the physics module and it was executed for these radial beam tube configurations on $30 \times 30 \times 25$ and $30 \times 30 \times 21$ Cartesian grids for the two alternative configurations, respectively. As before, these calculations employed an Upward-100 biased angular quadrature set, 26 energy groups, and $P_0$ scattering cross-section data. A material dependent source module was specifically developed for the first radial beam alternative configuration and used in the physics calculations for the both alternative model configurations.

The results of this optimization calculation are presented and discussed in the next section.
5.2.2.1 Results of MOZAIK with Min-max

For the radial beam tube configuration-I, MOZAIK with Min-max obtained an optimal tank shape in 123 optimization iterations. In contrast, for the second configuration, MOZAIK with Min-max obtained an optimal tank shape in 153 optimization iterations. The thermal neutron beam intensity computed with the source module, and for selected states with source distributions computed via the MCNP Core Model, for these two configurations are shown in Figs. 5.29 and 5.30. The very good agreement between the beam intensities with the two boundary source-calculation strategies for configuration-I in Fig. 5.29, and the reasonable agreement for configuration-II in Fig. 5.30, represent a verification of successful performance by the source module for these two radial beam configurations.

The results in Fig. 5.29 illustrate that radial beam tube configuration with this optimal tank shape produces neutron beam intensity at the beam tube exit end comparable to that produced by the configuration reported in Section 5.1. This is not surprising since these two model configurations have similar neutronic features, namely the neutrons coming from the core have to travel almost 11 cm through D$_2$O before reaching the beam re-entry hole. In other words, the neutron scalar flux at the beam tube base for these two configurations is almost same. In addition, the results of the black absorber experiments presented in Section 5.1.5 illustrate that the contribution of the forward scattering component of the exit thermal neutron partial current is dominant for both radial and tangential configurations. As a result, the tangential and radial configurations produce comparable thermal neutron beam intensities at the beam tube exit end.

The radial beam tube configuration-II, produces the highest thermal neutron output at the beam tube end among all considered configurations. We attribute this desirable result to the fact that in configuration-II the beam tube base is so close to the core interface that most core neutrons coming from core enter the beam re-entry gap directly, and their vast majority contribute to the thermal neutron beam at the beam tube exit end.

The optimal shape found by MOZAIK with Min-max for the first radial beam tube configuration is shown in Fig. 5.31. This shape was smoothed by the procedure described in Section 5.1 and an the smoothed optimal shape is shown in Fig. 5.31. The smoothed shape comprises a cylinder, or drum shape, that covers a
Figure 5.29. Thermal neutron beam intensity at the exit hole of the best state per search iteration as computed by MOZAIK with Min-max for the radial beam tube configuration-I

Figure 5.30. Thermal neutron beam intensity at the exit hole of the best state per search iteration as computed by MOZAIK with Min-max for the radial beam tube configuration-II
portion of the beam re-entry gap, whose radius is 19 cm and height is 24.5 cm.

For the second radial beam tube configuration, the obtained optimal shape, and its smoothed version are shown in Fig. 5.32. In this model configuration, the verifier module smoothed the shape and produced a conical shape tank. For the optimal shape determined by MOZAIK with Min-max, Fig. 5.32, the neutron partial current in the thermal group exiting the beam tube was computed as $7.88 \times 10^{-9} \text{n/cm}^2\text{-s}$. The same quantity was computed for the smoothed shape of configuration-II as $8.67 \times 10^{-9} \text{n/cm}^2\text{-s}$. Again the 9.3% difference between the results of these two models (with/without smoothing) is due to the coarse mesh structure employed in TORT during the optimization calculations.

Similarly, for the first radial beam tube configuration, the thermal neutron beam intensity for the optimal shape and for the smoothed optimal shape are $4.88 \times 10^{-9} \text{n/cm}^2\text{-s}$, and $5.63 \times 10^{-9} \text{n/cm}^2\text{-s}$, respectively.

Figure 5.31. 3D-views of the optimal D$_2$O tank shape and its smoothed version for the radial beam tube configuration-I

Figure 5.32. 3D-views of the optimal D$_2$O tank shape and its smoothed version for the radial beam tube configuration-II
5.3 Summary of MOZAIK’s Performance

In this chapter, four different optimization problems are described and their results obtained with MOZAIK are summarized. Table 5.5 summarizes the results for the optimization problems with different beam tube arrangements.

Table 5.5. Outgoing partial current per energy per fission event in the thermal group computed at the beam tube exit end by TORT and the streaming operator for four beam tube arrangements.

<table>
<thead>
<tr>
<th>Shape</th>
<th>Tangential (Existing BT)</th>
<th>Tangential (near the core)</th>
<th>Radial BT - I</th>
<th>Radial BT - II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td>4.83 e-9</td>
<td>6.12 e-9</td>
<td>4.88 e-9</td>
<td>7.88 e-9</td>
</tr>
<tr>
<td>Smoothed</td>
<td>5.84 e-9</td>
<td>7.13 e-9</td>
<td>4.63 e-9</td>
<td>8.67 e-9</td>
</tr>
</tbody>
</table>

The results show that:

1. Radial beam tube configuration-II is the best configuration. In this configuration, the beam tube base is placed close to the core interface, and the beam tube axis is normal to the core face. The thermal neutron beam intensity at the beam tube exit end increases by a factor of 1.8 for this configuration compared to the existing configuration’s $4.41 \times 10^{-9}$ value. It is worth noting here that the goal of the optimization exercise here was limited to maximizing the thermal neutron beam intensity at the exit hole. In reality, experimentalists would also demand a low gamma beam intensity, which typically tends to prefer a tangential beam arrangements.

2. The tangential beam tube configuration in which the beam tube is placed near the core interface is the second best model. It improves the thermal neutron beam intensity by a factor of 1.4 compared to the existing beam tube arrangement.

3. In all configuration, the volume of D$_2$O added by MOZAIK to the search problem domain is always smaller than the volume of D$_3$O in the current beam tube configuration. This make it easier to arrange more than one beam tube around the core’s faces.
4. The primary task of the D$_2$O in the system is reflecting thermal neutrons back into the core rather than moderating fast neutrons escaping the core into the thermal range before streaming into the beam tube. Therefore, MOZAIK’s optimizer module first prefers placing D$_2$O in the region around the beam tube base and beam re-entry gap within the search problem domain as these locations contribute more significantly to the thermal neutron beam intensity at the exit hole.

5. Model-splitting, which reduces the computational time significantly, is possible if a good bootstrapping strategy is developed for the coupling of the separated models. In our case the ability to accurately and efficiently determine the detailed boundary source enables the model-splitting. However, our results show that the specifics of the successful source generation approach are model dependent. One source generation strategy developed for one configuration cannot be utilized for the other models.

6. MOZAIK with both optimizers is operational and successful in determining the optimal shape of the D$_2$O tank. The usage of the two different optimization techniques, Min-max and Genetic Algorithms, also shows that MOZAIK’s modular feature is successfully achieved.

7. Parallel performance of MOZAIK makes it preferable for the similar applications.
Conclusion and Future Research

6.1 Conclusions

In a few decades, using robust techniques to deal with geometric optimization problems has become more popular due to advances in the computer technology. Employing such modern robust techniques for solving shape optimization problems is essential for the purpose of nuclear engineering applications, since for these applications, the optimization procedures applied are presently manual in nature and limited in scope.

The modular optimization code package, MOZAIK, was designed to address geometric optimization problems involving radiation transport in nuclear engineering applications. Oak Ridge National Laboratory’s TORT code was selected as the transport solver in the physics module of MOZAIK, and two different optimizers, Min-max and Genetic Algorithms, were provided in the optimizer module of the code package. MOZAIK’s first mission was to determine the optimal shape of the D_{2}O moderator tank for the current and new beam tube configurations for the Penn State Breazeale Reactor’s (PSBR) beam port facility.

A set of computational models based on the existing beam port # 4 configuration of the PSBR was designed to establish for the physics code the parameter range that yields accurate, computationally efficient solutions to the transport equation, before starting the optimization calculations for the beam port facility. In addition, these computational models were used to test and validate the modules of MOZAIK separately and the whole code package prior to the actual...
optimization calculations. At the same time, the tools necessary specifically for the optimization calculations of the PSBR were developed and tested with these models. The selected physics code, TORT, and the requisite data such as source distribution, cross-sections, and angular quadratures were comprehensively tested with these computational models.

One of the most important outcomes of the model development studies is that it is impractical to compute the neutron beam intensity at the beam tube exit end using a full model involving the PSBR’s core, moderator tank, and whole beam tube. In contrast, a model-splitting approach with a good bootstrapping strategy is a practical mechanism to compute the exit beam intensity accurately at a reasonable computational cost. The results indicate that the shorter computation time enables usage of TORT as a transport solver in the physics code for the modular optimization code package to design the new beam ports for the PSBR in a reliable and accurate way. Because of the poor statistics of the tallied result, using MCNP without biasing techniques is not suitable for our purposes. Moreover, the results show that the ANSL-V cross-section set is suitable for the PSBR’s beam port design calculations with MOZAIK.

The results also indicate that the D$_2$O moderator tank in the existing beam port configuration of PSBR was over-designed. In addition, the computed neutron energy spectra imply that the primary function of the heavy water is reflection rather than moderation of the neutrons leaking out of the core.

The developed streaming operator provides the neutron beam intensity at the beam tube exit end without performing a transport computation along the entire length of the beam tube. The TORT Tank Model with the computed boundary source and streaming operator work well together to compute the fluxes/currents at the beam tube exit end consuming (reasonable computation times of approximately 30 min and 2 min, respectively).

Multiprocessing was implemented to reduce the execution time MOZAIK consumes primarily in executing the physics calculation. A distributed memory parallelism with dynamic scheduling was applied to MOZAIK via MPI to execute the physics module for various states in the same search iteration on a number of processors concurrently. In this way, the total computation time consumed by the physics module is significantly reduced. This also encourages the usage of
MOZAIK for shape optimization problems in nuclear applications because many codes related to radiation transport do not have parallel execution capability. Observed performance indicates that the parallel version of MOZAIK is operational and yields speed-up at reasonable parallel efficiency on 32 processors in a homogeneous multiprocessor environment.

A method was introduced to reduce the extent of the search problem domain of the optimization calculation performed on one of the beam tubes of the PSBR. The results show that D$_2$O is the best moderating material for this configuration, compared to H2O, air, and graphite, and that the largest fraction of the exiting beam neutrons originates from the forward scattering component. These results show that the preferred region for placement of a reflecting material is around the beam tube base in the search space. The results also verify conclusions reached from preliminary optimization calculations where the optimizer computed the optimal D$_2$O tank shape as a hemi-sphere around the beam tube base. Consequently, by developing this methodology, limits on the extent of the search problem domain that directly affects the computation time of the optimization algorithm were determined and used in conducting the actual optimization calculations.

As a consequence of preliminary testing results it was concluded that MOZAIK is operational with two different optimizers and that it can be applied for the design calculation of the new beam ports of the PSBR’s beam port facility. First the optimal tank shape of the current beam port configuration was sought by MOZAIK. Next, three new alternative configurations were examined with different beam tube arrangements. Then the optimal tank shape for these new models were sought by MOZAIK. The results of the optimization exercise show that the radial beam tube configuration whose base is 3 cm away from the core interface, with its optimal D$_2$O tank shape is the best performing configuration, producing a thermal neutron beam intensity almost 1.8 times larger than with the existing beam tube design. In general, this study suggest that moving the beam tube closer to the core interface increases the neutron beam intensity at the beam tube exit end for both the tangential and radial beam tube configurations. All alternative arrangements show that higher neutron beam intensity can be obtained at the beam tube exit end using a smaller volume of D$_2$O in the system. This enables individual moderator chamber design for each beam tube if several beam tubes are being planned to
place around the reactor core.

All these results indicate that MOZAIK is viable and effective and it ready for deployment to address shape optimization problems involving radiation transport in nuclear engineering applications. Its modular feature and multiprocessing capability encourage the usage of MOZAIK for shape optimization problems in nuclear applications where many radiation transport codes lack parallel execution capability.

6.2 Future Research

In the calculations described in this work, the optimal tankshape was sought that only maximizes the thermal neutron beam intensity while minimizing the fast neutron intensity at the beam tube exit end. In other words, the cost function was defined using a simple objective. For other applications a suitably defined multi-objective cost function definition should be provided to the optimizer module of the code package. In this way, two or more quantities are evaluated when determining the optimal shape, (e.g. seeking a moderator/ reflector shape to produce both a higher thermal neutron beam intensity and a spatially uniform neutron beam).

Similarly, a strategy could be developed to optimize several beam tube configurations at the same time. In this case, separate search problem domains for each beam tube are designed. The optimization calculation is started for one of the beam tubes until a few search iterations are completed. Then, the neutron boundary source from the selected beam tubes problem domain into the other problem domains is updated only if there is substantial interference between them. After this updating process, optimization calculations are conducted for the next beam tube. This procedure is repeated for all beam tubes until one of them is converged. By removing this converged problem from the calculation sequence, the optimization calculations are continued for the other beam tubes. The calculations are stopped when all individual beam tube shapes are converged (determined an optimal shape).

Since the search is usually applied on a sufficiently coarse mesh to reduce the total execution time during the optimization procedure, the determined optimal results should be carefully smoothed to a final shape which can be realistically
constructed. For this purpose, a robust smoothing algorithm must be developed.

This code package can be applied for a broad variety of problems including optimal component shape design, e.g. optimization of size and shape of the cold moderator chamber for a cold-neutron source, re-entry gap, radiation filters, collimators, and reflector blocks for research reactor applications. Moreover, it can be provided for the source-detector geometry optimization which is widely used in nuclear non-proliferation and homeland security applications.
Bibliography


Kursat B. Bekar was born in Erzurum, Turkey on February 5th, 1974. He received his Bachelor degree in Nuclear Engineering from the Department of Nuclear Energy Engineering, Hacettepe University in June 1996. He started to work as a research and teaching assistant at the same department. He earned his M.S. degree in Nuclear Engineering from Hacettepe University in December 1999. He started to pursue his PhD degree in Nuclear Engineering at Penn State University in May 2004. He received his PhD degree in August 2008. During his PhD study, he also worked as a graduate research assistant at the Department of Mechanical and Nuclear Engineering, Penn State University.