

FY2010 Status Report on Advanced Neutronics Modeling and Validation

Nuclear Engineering Division

About Argonne National Laboratory

Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory's main facility is outside Chicago, at 9700 South Cass Avenue, Argonne, Illinois 60439. For information about Argonne, see <http://www.anl.gov>.

Availability of This Report

This report is available, at no cost, at <http://www.osti.gov/bridge>. It is also available on paper to the U.S. Department of Energy and its contractors, for a processing fee, from:

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
phone (865) 576-8401
fax (865) 576-5728
reports@adonis.osti.gov

Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor UChicago Argonne, LLC, nor any of their employees or officers, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of document authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, Argonne National Laboratory, or UChicago Argonne, LLC.

FY2010 Status Report on Advanced Neutronics Modeling and Validation

**M. A. Smith, A. Mohamed, A. Marin-Lafleche, E. E. Lewis, K. Derstine, C. H. Lee,
A. Wollaber, and W. S. Yang**
Nuclear Engineering Division
Argonne National Laboratory

September 15, 2010

SUMMARY

UNIC is the neutronics component of the massively parallel, multi-physics SHARP (Simulation for High-efficiency Advanced Reactor Prototyping) framework under development at Argonne National Laboratory. During this fiscal year, the SN2ND solver, MOCFE solver, and NODAL solver received significant development to meet the needs of the SHARP project. Additional follow-on analysis of the ZPR-6/6A from the previous year was performed in addition to new analysis of the ZPR-6/7 experiments where UNIC predictions of ZPR foil activation were made against experimentally measured values.

The SN2ND solver was applied to a plate-by-plate model of ZPR-6/6A using 294,912 cores of BlueGene/P and 222,912 cores of XT5, the two largest open-science high performance computing machines. The calculations proved that the SN2ND solver could be applied to heterogeneous reactor modeling problems, but more solver development (e.g., multigrid preconditioner) and computing power are required before such calculations are routine. As a consequence, the SN2ND solver was revised to incorporate a new multigrid preconditioner concept in addition to removing the remaining inappropriate spherical harmonics related quantities left over from its beginning (i.e. PN2ND). At the time of this report, the new version of SN2ND has not been completed, and the follow on work for SN2ND will continue to focus on updating the preconditioner as outlined in this report.

The MOCFE solver was rebuilt into UNIC in the previous year such that it obeyed the basic concepts of parallelism (scalable memory and communication). The MOCFE parallel algorithm was fully debugged this year and initial scalability tests on over 2048 processors were carried out such that the parallel algorithm could be assessed. That work indicated that a significant load imbalance in the coefficient matrix-vector application exists. A possible solution was formulated, but it has not been fully tested at the time of this report. Various setbacks caused by numerous ray tracing problems and a mistake in the implementation were unanticipated thus delaying progress on the MOCFE solver and the targeted development tasks for MOCFE were not completed this year.

In addition to the high fidelity solvers SN2ND and MOCFE, some time was spent implementing the NODAL solver. NODAL is similar to an existing legacy tool, but employs parallelism for enhanced performance and a capability to map a heterogeneous geometry into the homogenized geometry. This solver would provide a path to improve upon the existing homogenization approaches used for fuel cycle analysis, transient analysis, and perturbation theory calculations. An appropriate preconditioner was identified for NODAL this year and the solver algorithm was partially completed in UNIC. To facilitate the validation tests of UNIC using the ZPR-6 critical experiments, the BuildZPRmodel tool was also updated and a mesh merging algorithm was created. In addition to the newly implemented “solution along a line” analysis capability, these tools proved crucial to being able to perform the foil activation analysis detailed in this report.

Combined tests of MC²-3 and UNIC were performed against ZPR-6/7 experiments. The as-built core models of four core loadings with high Pu-240 zone (Loading 104, 106, 120, and 132) were analyzed by modeling more than 100 drawer types explicitly. Cell-averaged drawer cross sections were generated using the 1-D transport capability of MC²-3 based on the ENDF/B-VII.0 data. The results indicated that MC²-3/UNIC performed very well on the

range of fast reactor problems consistent with the cross section generation procedure implemented in MC²-3. For all the four core loadings analyzed, the core reactivity was predicted within 1- σ (standard deviation) of the estimated experimental uncertainty (~80 pcm), including the geometry and composition uncertainties. This result is comparable to the accuracy of MCNP Monte Carlo solutions; the UNIC solutions deviated from the measured values by 75, 43, 28 and -24 pcm for the Loadings 104, 106, 120, and 132, respectively, while the corresponding deviations of MCNP solutions were -56, -42, -132, and 0 pcm.

For the Loadings 104 and 120 of conventional fast reactor compositions, the calculated reaction rate distributions for enriched uranium fission, depleted uranium capture and fission, and plutonium fission agreed well with the foil activation measurements within 1- to 2- σ of the measurement uncertainties. However, for the Loadings 106 and 132, which have BeO plates around the central sodium drawer, more than 3- σ deviations were observed for the depleted uranium capture reaction rates near the BeO plates. Additional changes to the cross section generation algorithm utilizing the existing MC²-3 solver with some enhancements (two- or three-dimensional MOC) should resolve the remaining issues.

TABLE OF CONTENTS

Summary.....	i
Table of Contents.....	iv
List of Figures.....	vi
List of Tables.....	vii
1 Introduction.....	1
2 SN2ND Development.....	3
2.1 Algorithmic Review.....	4
2.2 Development of a Multi-grid Preconditioner.....	7
2.3 Implementation Issues with Parallelization in Energy.....	10
3 MOCFE Development.....	12
3.1 Review of MOCFE Solution Scheme.....	12
3.2 Problems Encountered in Ray Tracing.....	14
3.3 A Scalable Back Projection Algorithm.....	21
3.4 Automatic Optimization Adjustor Algorithm for MOCFE.....	23
3.5 Initial Scalability Results for MOCFE.....	29
4 NODAL Solver Development.....	32
4.1 Construction of the NODAL Preconditioner.....	32
4.2 Comparison of GMRES to Existing VARIANT Algorithm.....	33
4.3 Orthogonalized Matrix Aggregation.....	35
4.4 Summary Discussion.....	36
5 Miscellaneous Components of the Neutronics Work.....	37
5.1 IsoNXS Storage of Tabulated Cross Section Data.....	37
5.2 Gauss-Lobatto-Tchebychev Finite Element Capability.....	39
5.3 Evaluating the Solution along a Traversing Line.....	40
5.4 BuildZPRmodel and MergeMesh, Programs to Generate Input Geometry.....	45
5.5 New Repository and Component Verification of UNIC.....	49
6 Verification and Validation Tests.....	51
6.1 Follow on Calculations of ZPR-6/6A.....	52
6.2 Space-angle Convergence Study of PN2ND and SN2ND.....	54
6.3 ZPR-6 Assembly 7 Experiment.....	56
6.3.1 Slab Geometry Scoping Studies.....	56
6.3.2 Eigenvalue Results for the Homogenized Drawer Calculations.....	57
6.3.3 Foil Results for the Homogenized Drawer Calculations.....	61
6.3.4 Follow on Work to Investigate the BeO Related Errors.....	63
7 Conclusions.....	65
8 References.....	67

LIST OF FIGURES

Figure 2.1 Serendipity (upper) and Lagrangian (lower) Quadrilaterals in UNIC.....	8
Figure 2.2 Tessellated Serendipity (upper) and Lagrangian (lower) Quadrilaterals in UNIC.....	9
Figure 2.3 Tetrahedral Tessellated Serendipity Wedge in UNIC	9
Figure 3.1. Coordinate System for the Characteristic Lines	13
Figure 3.2. Example Decomposition of the Mesh and Segmentation of Crossing Trajectories	15
Figure 3.3. Example MOCFE Ray Tracing Problem on the Domain Surface.....	16
Figure 3.4. Example MOCFE Ray Casting Problem on the Domain Interior	17
Figure 3.5 Example Segmented Ray Tracing Data.....	18
Figure 3.6 Example Four-Processor Back Projection.....	22
Figure 3.7 Example Error Measures for the 33 Group ABTR Test Problem	25
Figure 3.8 Example Dominance Ratio (DR) Impact on Solution Convergence.....	26
Figure 3.9 Within-group Results for the VHTR without Group Balancing	27
Figure 3.10 Within-group Results for the VHTR with Group Balancing.....	27
Figure 3.11 Example Multi-Domain Decomposition Approach to the VHTR.....	31
Figure 4.1 Quarter Core Configuration of Six LWR Fuel Assemblies and Reflector	34
Figure 4.2 Convergence Rates of Red-Black Gauss-Seidel and GMRES for NODAL	34
Figure 4.3 Impact of Combining p -multigrid and Orthogonalized Matrix Aggregation	36
Figure 5.1 One-Dimensional GLT and Lagrangian Finite Element Basis Functions.....	40
Figure 5.2 VISIT Sub-element Approximation of a Finite Element	41
Figure 5.3 Quadratic and Quadric Finite Elements and a Flux Traverse (x 's).....	41
Figure 5.4 Assumed Quadratic and Quadric Flux Solutions	42
Figure 5.5 Solutions and Calculated Error of the Sub-element Scheme.....	42
Figure 5.6 ZPR-6/7 Matrix Edge Group 1 Flux Plot for Loading 106 (Mesh Convergence)...	44
Figure 5.7 ZPR-6/7 Center Drawer Group 1 Flux Plot for Loading 106 (Mesh Convergence)	44
Figure 5.8 ZPR-6/7 Matrix Edge Group 1 Flux Plot for Loading 106 (Ray Effects).....	45
Figure 5.9. Single ZPR-6 Core Drawer	46
Figure 5.10. Example Homogenization Models	47
Figure 5.11. Full Core ZPR-6/7 Model Built with BuildZPRmodel	48
Figure 5.12. Example VHTR Mesh Built Using the New Merge Mesh Routine	50
Figure 5.13. Example Input Specification for Merge Mesh Routine in UNIC.....	51
Figure 6.1. ZPR-6/6A Fast Flux (left) and Power (right)	53
Figure 6.2. Space-angle Mesh Convergence Study Test Problem.....	55
Figure 6.3. Representative ZPR-6 Assembly 7 Core Drawer Layout.....	56
Figure 6.4. ZPR-6 Assembly 7 Loadings 104, 106, 120 & 132	58

Figure 6.5. ZPR-6 Assembly 7 Selected Flux Plots for Loading 106.....59
 Figure 6.6. Foil Locations in the ZPR-6 Assembly 7 High ²⁴⁰Pu Loadings62

LIST OF TABLES

Table 3.1 Preliminary Strong Scaling in Space Numbers for MOCFE29
 Table 3.2 Variation in the Number of Elements and Trajectory Data30
 Table 4.1 Calculated Performance Gain Using Orthogonalized Matrix Aggregation36
 Table 5.1. IsoNXS File Description: FileWide Group.....38
 Table 5.2. IsoNXS File Description: Isotope Header Group (repeated for each Isotope)38
 Table 5.3. IsoNXS File Description: Isotope Reaction Groups (repeated for each Isotope)38
 Table 6.1. SN2ND Eigenvalue Error for the Drawer Homogenized ZPR Model.....52
 Table 6.2. Weak Angle Scalability of SN2ND on BlueGene/P (combined ANL and JSC).....52
 Table 6.3. SN2ND Eigenvalue Error (pcm) Energy Convergence for Two Different
 Meshes53
 Table 6.4. SN2ND Eigenvalue Error (pcm) Angular Convergence.....54
 Table 6.5. PN2ND Results for the Small Test Model with Reflective Boundary
 Conditions55
 Table 6.6. PN2ND Results for the Large Test Model with Reflective Boundary
 Conditions55
 Table 6.7. PN2ND Results for the Large Test Model with Vacuum Boundary Conditions.....55
 Table 6.8. MC²-3 k_∞ results for a Single Fuel Drawer of the ZPR-6/7, Loading 10457
 Table 6.9. Eigenvalue Solutions for ZPR-6 Assembly 7 Loading 104.....60
 Table 6.10. UNIC and MCNP Eigenvalues for Loadings 104, 106, 120 and 132.....61
 Table 6.11. Foil Reaction Rate Comparisons for Loading 104 and 106 (C/E-1 in %)62
 Table 6.12. Foil Reaction Rate Comparisons for Loading 106 with Explicit BeO Plates63
 Table 6.13. U-238 Capture Foil Reaction Rate Comparisons for Loading 106 with
 Modified Foil Cross Section Models64

1 Introduction

As part of the Advanced Fuel Cycle Initiative (AFCI), a fast reactor simulation program was launched in April 2007, termed SHARP [1], to develop a suite of modern simulation tools specifically for the analysis and design of sodium cooled fast reactors. The general goal is to reduce the uncertainties and biases in various areas of reactor design activities by providing enhanced prediction capabilities. Under this fast reactor simulation program, a high-fidelity deterministic neutron transport code named UNIC [2-3] was started. The application scope targeted for UNIC ranges from the homogenized assembly approaches prevalent in current reactor analysis methodologies to explicit geometry, time dependent transport calculations coupled to thermal-hydraulics and structural mechanics calculations for reactor accident simulations.

The creation of a single solver that can perform all of these calculations and be competitive with the wide range of analysis tools already in use is somewhat formidable, especially when considering the limited amount of manpower dedicated to this project. Given the large assortment of transport solvers capable of treating the assembly homogenized, if not pin-cell homogenized geometry, the initial focus of UNIC was to create new solvers appropriate for the multi-physics (structural mechanics and thermal-hydraulics) coupling problems of immediate interest that are beyond the modeling capabilities of any available solver today. Consequently, from its inception UNIC focused heavily on trying to discretize the transport equation preserving as much geometrical information as possible (i.e. little or no homogenization). With these guidelines, the primary development solvers, SN2ND and MOCFE, within UNIC have been researched for approximately 2.5 years with moderate success, however, given the large problem size associated with such problems ($>10^{14}$ degrees of freedom or unknowns), they can only be deployed on existing or future high performance computing machines.

Early in fiscal year 2010 (FY2010), a similar project focused on high fidelity modeling of the VHTR and other thermal reactor concepts, called NNR [4], was merged in with the SHARP project. While these two projects are conceptually different, they both contain good ideas for reactor analysis modeling and we are motivated to merge the concepts together to produce a single deployable set of tools. While not a targeted goal for this year, the most important aspect to take from the NNR related work is the need to provide a practical solution capability usable for most areas of reactor physics on much smaller computing resources. This is something that UNIC has not demonstrated a capability for thus far. Given that the projects are still separate; this report only considers the changes made to UNIC during the current fiscal year where continuing development of the high fidelity solvers dominates the workload. However, we note that some of the FY2010 work, NODAL, was intended to mimic the mindset of the NNR approach, but its purpose of course is for fast reactor analysis.

Most of the work done in FY2009 was spent improving the SN2ND solver performance and demonstrating its ability to treat a real heterogeneous geometry: ZPR-6/6A. In FY2010, we continued studying the ZPR-6/6A problem before spending any additional development time on SN2ND such that we could guarantee the underlying methodology (second order discrete ordinates) could provide an accurate solution capability to the heterogeneous problems of interest to SHARP. The conclusion from that work was that, given accurate cross section data and sufficient computing power, SN2ND can provide a viable solution capability

which does not currently exist with available tools. Thus additional development of SN2ND in FY2010 was justifiable.

In FY2010 we choose to investigate better multigrid methodologies which considered not only the space-angle system, the primary path of parallelism in SN2ND to date, but also the energy-space-angle system. Combined with the desire to model time dependent problems, this necessitated a considerable revision to SN2ND which we discuss in this report. We also carried additional verification and validation testing of SN2ND by studying the ZPR-6/7 experiments using homogenized drawer models. A partially dehomogenized methodology was also implemented in a scoping study along with a full blown heterogeneous plate-by-plate model although results of the latter are not included because of insufficient results. Comparisons using the homogenized models produced foil activation results consistent with the experimentally measured values and can generally be considered quite good. The calculations that were inaccurate were studied in more depth and identified to be limited by the cross section processing methodology which should be resolvable with further development of the cross section generation procedure.

Significant time in FY2009 was also spent rebuilding the MOC solver such that it incorporated the essentials of a scalable algorithm (scalable memory and communication). In FY2010, additional MOCFE development was pursued to test the current parallel algorithm on the large scale parallel machines. Numerous problems were identified with the parallel ray tracing algorithm in addition to a mistake in the implemented formulation which delayed the progress on MOCFE deployment. As an example, part of the FY2010 goal was to incorporate the two- and three-dimensional modeling capabilities of MOCFE into the MC²-3 [5-6] code. While the specific details on that work are provided in a companion report, we only mention it here because the delays in MOCFE prevented that work from being completed this year. While we were able to create a reliable solver this year, one which should satisfy the needs of MC²-3, the parallel performance is quite poor and requires follow on development as outlined in this report.

Given the prevalence of spatial homogenization in conventional reactor analysis combined with the resource requirements of the preceding MOCFE and SN2ND solvers, we desire a capability within UNIC to start at the current level of reactor analysis and transition smoothly (i.e., with familiar input/output) to the more explicit geometry modeling. We therefore chose to spend some time in FY2010 on the NODAL solver capability started in the previous year. This solver duplicates a legacy tool, VARIANT [7-8], but the added capability of parallelism in energy and ability to map a heterogeneous geometry into the homogenized geometry should provide a path to improve upon the existing homogenization approaches used for fuel cycle analysis, transient analysis, and perturbation theory calculations. We pursued this solver because we cannot expect an unstructured mesh-based solver such as PN2ND, SN2ND, or MOCFE to be computationally competitive with solvers that exploit structured geometry mesh descriptions as demonstrated in the last few years.

Besides the various solvers in UNIC, we also had to exert a considerable amount of effort building auxiliary components. Tools such as IsoNXS and “Solution along a line” have been discussed previously and the actual implementation details are either revisited or discussed in this report. One particular project task, ZPRtoNODAL, required a considerable amount of development time in FY2010 such that the ZPR related tasks could be completed. As discussed in previous reports, the mesh generation tools available are not necessarily reliable

or usable for complicated 3-D geometries. The ZPR geometries themselves pose a significant meshing problem because of the simple, yet tedious geometry combined with the definition of a large number of material regions.

In FY2009 we spent a significant amount of time building the ZPRtoNODAL tool which creates a structured geometry model of a ZPR with the material mapping naturally embedded. After working on ZPR-6/6A, we determined that no ZPR can be modeled properly using a structured geometry definition and we invested further effort this year generalizing the tool to create an unstructured mesh, renaming it BuildZPRmodel. We also added the capability to produce the MC²-3 input decks associated with the user-specified homogenization model for every ZPR calculation. Thus, with relatively simple input, we can now generate anything from a drawer homogenized model of a ZPR (inherently a structured geometry problem) to a full plate-by-plate model where the geometry, cross section generation input, and material mapping are all handled cleanly by the tool. This tool was exclusively used to create all of the ZPR-6/7 related input carried out this year and will require additional verification on the same level as what we have dedicated to the solvers in UNIC.

This report is organized into sections that discuss each of the above cited topics. Section 2 discusses the work carried out on improving SN2ND during FY2010 and discusses the remaining work yet to be completed. Section 3 discusses the MOCFE work, the numerous ray tracing problems and their solutions, and gives some detail on the initial scalability study. Section 4 discusses the work done on the NODAL solver and Section 5 details the auxiliary tools which were developed to support the various goals and tasks of UNIC in FY2010. Section 6 summarizes the benchmarking calculations tackled with UNIC in FY2010 which will form the basis of further verification and validation testing of the neutronics package we are building.

2 SN2ND Development

The SN2ND solver, originally started in mid-2007, can be safely stated to be the only successful solver created within UNIC with respect to parallelism and performance. The achievements of this solver are made possible by partitioning the space-angle system of equations over the available processors and utilizing established iterative solution techniques from the neutron transport community combined with the modern algorithms in the PETSc toolbox [9]. In FY2009, several areas were targeted in SN2ND for improvement such that its ability to tackle larger problems was improved and the time-to-solution was reduced. This effort was focused on proving the ability of the SN2ND method to treat heterogeneous problems which was partially demonstrated on the ZPR-6/6A in FY2009 although accuracy was a significant issue because of insufficient computing power.

From its inception, SN2ND utilized many parts of the PN2ND solver, around which UNIC itself was built, which are not appropriate for SN2ND since it utilizes a discrete ordinates angular approximation and PN2ND utilizes a spherical harmonics one. As UNIC has been generalized, so has SN2ND, and while the initial version of SN2ND was partially rewritten in FY2008 to accommodate the preconditioner that allowed it to progress from a less than 3,000 processor capability to a greater than 100,000 capability, the FY2008 and FY2009 version still relied heavily upon the PN2ND components, specifically the vector storage algorithm, to minimize code development efforts. While this choice does not affect the steady state eigenvalue calculations that SN2ND has been used upon thus far, time

dependent and adjoint calculations fundamentally require the source vectors in SN2ND to exist in the S_N space rather than the P_N space and thus SN2ND cannot be used in either regime without substantial changes. Two other factors also motivate making improvements to SN2ND: multi-grid preconditioner development and parallelism in energy. Given that MOCFE has yet to prove a reasonable performance capability (in both accuracy and computational efficiency) like that of SN2ND, we are still motivated to continue development of SN2ND such that we can accommodate the short term needs of NEAMS.

While any one of these issues could have been incorporated into the existing coding of SN2ND with moderate to considerable difficulty, incorporating all three necessitated a significant investment in reorganization of the SN2ND solver and the definition of new data structures and algorithms that are specifically tuned for SN2ND solver development. While the current version of SN2ND is still usable and in fact was the sole version used to obtain the results in this report, it has effectively reached its development end and the FY2010 work was entirely absorbed in rebuilding SN2ND which at the time of this report remains to be fully completed. The following sections detail the issues with changing the formulation to accommodate parallelization by energy, followed by the development of new algorithms for a spatial multi-grid preconditioner.

2.1 Algorithmic Review

The desire to implement parallelism in energy requires significant changes to the SN2ND solver. To begin, we revisit the even- and odd-parity within-group transport equations

$$\hat{\Omega} \cdot \vec{\nabla} \psi_g^-(\vec{r}, \hat{\Omega}) + \Sigma_{t,g}(\vec{r}) \psi_g^+(\vec{r}, \hat{\Omega}) = W_g^+(\vec{r}, \hat{\Omega}) + S_g^+(\vec{r}, \hat{\Omega}) \quad (2.1)$$

$$\hat{\Omega} \cdot \vec{\nabla} \psi_g^+(\vec{r}, \hat{\Omega}) + \Sigma_{t,g}(\vec{r}) \psi_g^-(\vec{r}, \hat{\Omega}) = W_g^-(\vec{r}, \hat{\Omega}) + S_g^-(\vec{r}, \hat{\Omega}), \quad (2.2)$$

where $\psi_g^\pm(\vec{r}, \hat{\Omega})$ is the even (+) and odd (-) parity flux and $W_g^\pm(\vec{r}, \hat{\Omega})$ is the within group scattering source and $S_g^\pm(\vec{r}, \hat{\Omega})$ are the out of group scattering and fission sources. The primary focus of the SN2ND derivation is to recast these two first-order equations into a second-order equation for the even-parity flux written in the discretized functional form:

$$T_e \left[\tilde{\psi}_{g,e}^+, \tilde{s}_{g,e}^\pm \right] = A_{g,e} \tilde{\psi}_{g,e}^+ - \tilde{s}_{g,e}^+ - \tilde{s}_{g,e}^- + BC_{g,e}. \quad (2.3)$$

Equation 2.3 is assembled over each finite element (e) in the domain and all angles in the cubature yielding a symmetric positive-definite coefficient matrix $A_{g,e}$ that is solvable using the efficient conjugate gradient (CG) algorithm as discussed previously [2].

Equation 2.3 is itself part of a larger iterative system which needs to be displayed to fully understand the implications of parallelism in energy and spatial multi-grid. We begin with the power method written as

$$A\Psi = \frac{1}{k} F\Psi = \frac{1}{k} Q \leftrightarrow \Psi = \frac{1}{k} A^{-1}Q \rightarrow Q^{i+1} = \frac{1}{k^i} FA^{-1}Q^i \quad (2.4)$$

where Ψ represents the flux vector assembled over space-angle-energy, A represents the assembled multi-group coefficient matrix which includes the within group matrices from equation 2.3 and group-to-group scattering terms, and i is the iteration index. The matrix F represents the fission source operator and k and Q are the fundamental eigenvalue and eigenvector which is all that is needed for most steady state calculations. As seen in equation

2.4, the power method is an iterative technique used to generate the fundamental eigenvalue and eigenvector, where the additional equation

$$k^{i+1} = k^i \frac{\|Q^{i+1}\|_1}{\|Q^i\|_1}, \quad (2.5)$$

is used to update the eigenvalue and complete the system.

Given that A is not directly invertible practically, an iterative inversion technique is used to approximate the inverse in equation 2.4. In the version of SN2ND without parallelism in energy, Gauss-Seidel (GS) is used in to solve equation 2.4 which can be written as

$$A\Psi = b \rightarrow (L+D)\Psi^{j+1} = b - U\Psi^j \rightarrow \Psi^{j+1} = (L+D)^{-1} [b - U\Psi^j]. \quad (2.6)$$

As can be seen, the coefficient matrix is factored into strictly lower triangular L , strictly upper triangular U , and diagonal D components. The inversion of $(L+D)^{-1}$ is merely the sequenced in group solution of each within group equation where latent upscattering components U are used to define the within group source. With regard to parallelism in energy, GS is easy to show as a non-scalable algorithm for most reactor calculations of interest because of the physical properties of the scattering (lower triangular dominate). To overcome this limitation in UNIC, GS is replaced with the GMRES algorithm operating on the entire space-angle-energy system. In GMRES, the sequenced solution algorithm of GS is swapped with a series of scalable coefficient matrix-vector applications ($A\Psi^j, A\Psi^{j+1}, A\Psi^{j+2}, \dots$), a scalable vector orthogonalization, and ideally a scalable preconditioner. It is the last part that remains to be researched and can easily take upwards of 2-3 FTEs worth of development for the problems of interest to SHARP.

Continuing, we will assume that any preconditioner for equation 2.6 will require the solution of a within group equation. Assembling equation 2.3 and extracting the within group scattering source from the generic sources we can write the following coupled system of equations for SN2ND

$$\begin{aligned} A_g \tilde{\psi}_g^+ &= \sum_{g'} W_{g' \rightarrow g}^+ \tilde{\psi}_{g'}^+ + \sum_{g'} B_g^T W_{g' \rightarrow g}^- \tilde{\psi}_{g'}^- + q_g^+ + B_g^T q_g^- \\ \tilde{\psi}_g^- &= B_g \tilde{\psi}_g^+ + \sum_{g'} W_{g' \rightarrow g}^- \tilde{\psi}_{g'}^- + q_g^- \end{aligned} \quad (2.7)$$

$W_{g' \rightarrow g}^\pm$ represents the group-to-group scattering source contribution, B_g represents the within group contribution, and q_g^\pm is a fixed source (such as fission for each iteration of the power method). Adding the iterative indices “ k ” for the typical source iteration solution scheme (Richardson) we obtain the following system

$$\begin{aligned} {}^{k+1}\tilde{\psi}_g^+ &= A_g^{-1} \left\{ W_{g \rightarrow g}^+ {}^k\tilde{\psi}_g^+ + B_g^T W_{g \rightarrow g}^- {}^k\tilde{\psi}_g^- + \sum_{g' \neq g} W_{g' \rightarrow g}^+ \tilde{\psi}_{g'}^+ + \sum_{g' \neq g} B_g^T W_{g' \rightarrow g}^- \tilde{\psi}_{g'}^- + q_g^+ + B_g^T q_g^- \right\} \\ {}^{k+1}\tilde{\psi}_g^- &= B_g {}^k\tilde{\psi}_g^+ + W_{g \rightarrow g}^- {}^k\tilde{\psi}_g^- + \sum_{g' \neq g} W_{g' \rightarrow g}^- \tilde{\psi}_{g'}^- + q_g^- \end{aligned} \quad (2.8)$$

We neglect the acceleration equation in this derivation for brevity. For calculations assuming isotropic scattering, the second equation is unimportant and further, for linear anisotropic scattering, the second equation amounts to a minor amount of computational effort. However,

at and beyond P_1 scattering, we observed significant problems with strong scaling in angle because the odd-parity flux was stored in the P_N space rather than S_N , an artifact of the PN2ND solver. For time-dependent and adjoint calculations and to obtain good scaling when parallelization by angle is incorporated, the odd-parity flux must be stored in the S_N space.

As stated earlier, the bulk of work in SN2ND has focused on the inversion of the coefficient matrix A_g in equation 2.8. For this matrix, SN2ND uses a parallel CG algorithm over the entire space-angle system written compactly as

$$A_g \tilde{\psi}_g^+ = b_g^+ \quad (2.9)$$

which we have shown to be scalable on more than 200,000 processors. In FY2009, this part of the SN2ND solution scheme was modified to incorporate optimized matrix-vector operations and a p -multigrid algorithm. While the p -multigrid algorithm yielded a net negative improvement in performance, the substantial savings in memory made many calculations previously thought to be intractable, specifically calculations with hundreds of groups, possible. Unfortunately the implementation is quite messy and relies upon hard wired iterative limits throughout the implementation. This was found to cause significant convergence issues when the types of problems solved went beyond the problems studied during the development phase. To understand how multi-grid is implemented, we define a preconditioner step for equation 2.9 as

$$M_g^{-1} A_g \tilde{\psi}_g^+ = M_g^{-1} b_g^+ \rightarrow Y = M_g^{-1} \{b_g^+ - A_g \tilde{\psi}_g^+\} = M_g^{-1} X = 0 \rightarrow M_g Y = X. \quad (2.10)$$

In equation 2.10, we always define a preconditioner M_g that is representative of the coefficient matrix in equation 2.9, but typically much simpler. The matrix M_g is rarely directly invertible and thus some numerical inversion technique such as GS is used. In SN2ND, a CG algorithm is used over the entire space-angle system for equation 2.9 and its preconditioner M_g , neglects the coupling in angle (assuming it exists [2]) such that each angular subsystem $M_{g,n}$ can be solved simultaneously. The matrix $M_{g,n}$ is also symmetric positive definite and we again utilize a CG algorithm to perform the necessary inversion appearing in equation 2.10. With such a scheme, we can write

$$M_{g,n} Y_n = X_n \ \& \ (M'_{g,n})^{-1} \{X_n - M_{g,n} Y_n\} = 0 \rightarrow M'_{g,n} Y_n = X_n, \quad (2.11)$$

where $M'_{g,n}$ is a preconditioner for the CG iterative scheme involving $M_{g,n}$. One can easily see how this process can continue in a hierarchical fashion where each new CG level will generate a simpler and simpler preconditioner.

This is fundamentally the concept behind developing a multi-grid preconditioner where the spatial “grid” becomes progressively coarser and thus cheaper from a numerical inversion point of view. The reason this approach is scalable and efficient is that the coefficient matrix application at each level is a known scalable process for the second-order diffusive like system that can be written at each preconditioner level of $M_{g,n}$.

As mentioned, the SN2ND solver utilizes the parallel CG algorithm in PETSc to invert $M_{g,n}$ where parallel SOR is defined as the preconditioner $M'_{g,n}$. The p -multigrid work of FY2009 focused on introducing a single level to the existing scheme written as

$$\begin{aligned} M_{g,n} Y_n &= X_n \quad \& \quad Y'_n = (M'_{g,n})^{-1} \{X_n - M_{g,n} Y_n\} = (M'_{g,n})^{-1} X_n \\ M'_{g,n} Y'_n &= X'_n \quad \& \quad Y''_n = (M''_{g,n})^{-1} \{X'_n - M'_{g,n} Y'_n\} = (M''_{g,n})^{-1} X'_n \end{aligned} \quad (2.12)$$

where application of $M_{g,n}$ was switched from a non-zero stored matrix-vector operation performed in PETSc to a dense matrix-vector operation in SN2ND. Given a mesh with quadratic finite elements, the preconditioner $M'_{g,n}$ would have the same spatial definition but utilize linear finite element trial functions. While the old SN2ND solver focused on assembling $M_{g,n}$ for PETSc where it would apply parallel CG with a parallel SOR preconditioner $M'_{g,n}$, in the p -multigrid version the lower order system $M'_{g,n}$ is assembled and given to PETSc so that it can apply CG with a parallel SOR preconditioner at the $M''_{g,n}$ level.

2.2 Development of a Multi-grid Preconditioner

The primary reason for the poor performance of the p -multigrid preconditioner was due to the fact that the linear mesh does not span the vector space of the originating quadratic (or higher order) mesh. This fact necessitated the use of an inefficient Jacobi iteration scheme to ensure proper inversion of the targeted preconditioner system. Since this preconditioner was implemented as a quick fix to circumvent the memory related problems on BlueGene/P [22], and the Jacobi iteration was added as a quick fix to that, it was not entirely surprising that the new preconditioner performance was poor from both an implementation and theory point of view. Nevertheless, it was an instructive learning experience that proved reliance upon specific examples from the literature are not necessarily appropriate because the typical case study focuses on structured grids with fixed material properties that are not terribly heterogeneous such as that observed for a fuel assembly in a nuclear reactor or a explicitly represented ZPR drawer.

Because the literature is not clear on what preconditioner will work best for the SHARP related problems, the two primary authors of the SN2ND solver decided it was best to consult resident experts on multigrid methodologies [10] rather than attempt another implementation in SN2ND that duplicated the experiences with the first multigrid implementation. The discussions with various authors from a wide range of application interests, all transport, but different problem foci, all suggested using algebraic multigrid [11] on the system of equations generated by SN2ND at the $M_{g,n}$ preconditioner level. A detailed understanding of algebraic multigrid is beyond the scope of this report, and one only needs to understand that given a matrix $M_{g,n}$, a typical algebraic multigrid implementation will consider the numerical and parallel setup of the problem and devise a way to create a hierarchical preconditioner similar to equation 2.12 but with more levels. This is accomplished by using progressive factorizations of the starting matrix and thus the only real trick to using an algebraic multigrid preconditioner is that the memory load is roughly double that already required by PETSc in

SN2ND to solve $M_{g,n}$. Given that SN2ND is already memory restricted on HPC systems, we must consider parallelization by energy and a one-level, possibly two-level preconditioning before algebraic multigrid can be considered.

In FY2009, significant work was done studying the matrix-vector operations in SN2ND to see if there was a way to optimize the matrix-vector applications. During this work it was observed that linearly interpolated tetrahedral, triangle, and bar elements could all be applied using a single element coefficient matrix. Given that the Jacobian for linear elements is trivial to evaluate and requires little storage in itself, this translates to large factors of improvement in performance for matrix-vector operations compared with other elements such as hexahedrons. Combining this experience in the discussions, all parties agreed that, to alleviate the memory burden, the first level of the preconditioner for this system should be an element-wise decomposition into these three-fundamental types which we term a linear tessellation or *LT*-multigrid since it does not match conventional *p*- or *h*- concepts for multigrid.

To describe the *LT*-multigrid approach, we consider the two-dimensional elements in Figure 2.1 which shows linear through cubic order quadrilateral elements available in UNIC.

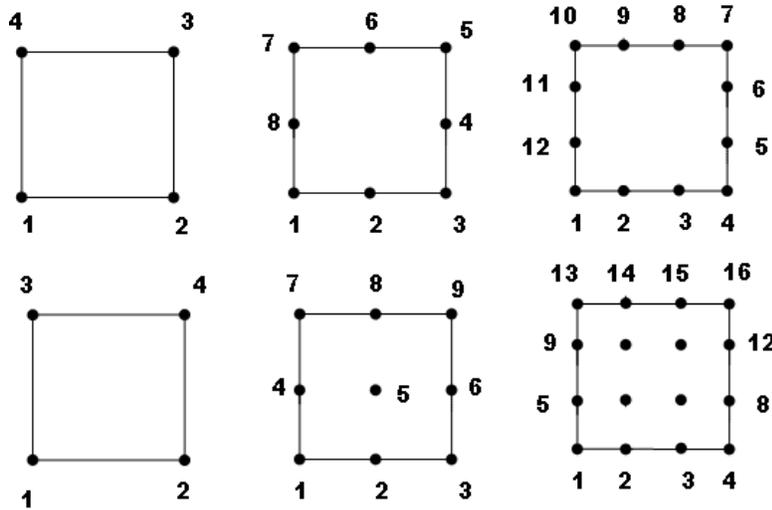


Figure 2.1 Serendipity (upper) and Lagrangian (lower) Quadrilaterals in UNIC

For the discretized equations on each element, the associated spatial matrices in Figure 2.1 are found to be fully connected. As an example, the global vertex connecting to the fifth vertex in the quadratic Lagrangian (lower middle) will have exactly 9 connections on the row corresponding to each vertex of that element. With respect to the global system of equations, assuming a structured mesh where each element is adjacent to at most 8 other elements, we can further state for the quadratic Lagrangian quadrilateral that each corner associated vertex (1,3,7,9) will have 25 connections per row and each side associated vertex (2,4,6,8) will have 15 connections. If we linearly tessellate this element as shown in Figure 2.2 and consider the same structured grid, we find that every vertex will have exactly 9 connections per row.

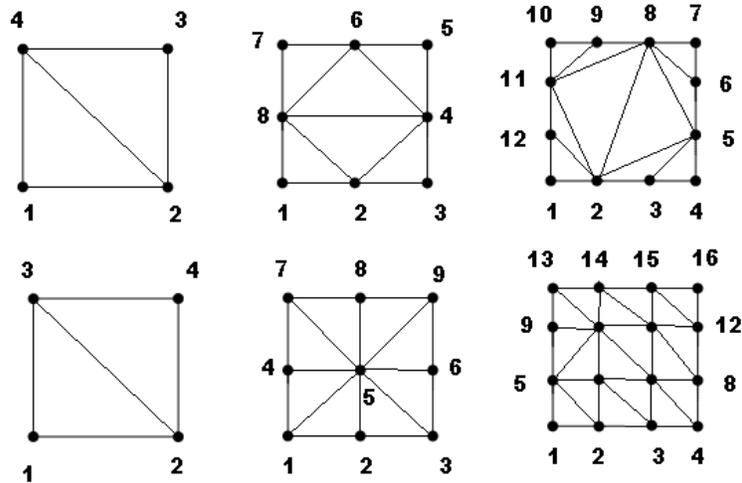


Figure 2.2 Tessellated Serendipity (upper) and Lagrangian (lower) Quadrilaterals in UNIC

With respect to assembling and storing a preconditioner $M_{g,n}$, this translates to approximately a factor of two savings in memory storage requirements over the quadratic element where the vector space is not changed. Note that as the order is increased from quadratic to cubic, etc., the memory savings are progressively higher. A similar trend is observable in 3-D elements where we decompose the elements into linear tetrahedrons as shown for the quadratic serendipity wedge (or prism) element in Figure 2.3.

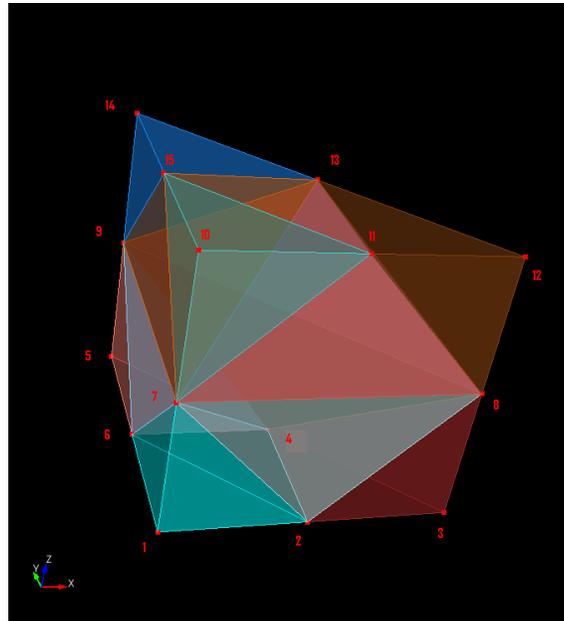


Figure 2.3 Tetrahedral Tessellated Serendipity Wedge in UNIC

For a quadratic Lagrangian hexahedral structured geometry mesh we anticipate factors of 2-3 reduction in memory for the assembled preconditioner where the other element types will generally produce less than the hexahedral case. This obviously reduces the memory burden considerably, but to utilize algebraic multigrid we would have to provide the assembled

system to PETSc. Given the estimated increase in memory requirements for algebraic multigrid (factor of two), we would expect a rather minor change in the current memory usage requirements of SN2ND and thus it would again not be practical on larger than 100 group problems on HPC machines such as BlueGene/P. As a consequence, we envision developing one further level of multigrid within SN2ND below the *LT*-multigrid step where we drop the higher order vertices on each finite element and create the linear subsystem.

This new level is structurally similar to the *p*-multigrid scheme of FY2009 where $M_{g,n}$ is composed of quadratic elements from Figure 2.2 and $M'_{g,n}$ is composed of the linear elements in Figure 2.2. Using the same structured grid model, we find that this reduces the vector space by a factor of two in addition to reducing the number of connections per row by a factor of two (a net reduction in memory of four). Unlike the old preconditioner, we propose using the linear tessellation of the fundamental elements shown in Figure 2.3 as an *h*-multigrid preconditioner $M''_{g,n}$ of the *LT*-multigrid preconditioner system $M'_{g,n}$. Since the vector space is not preserved, we are required to use the Jacobi iteration scheme again or modify the preconditioner definition such that it does appropriately span the higher order system. While we have some ideas on how to implement this preconditioner without needing a Jacobi iteration scheme, this research will obviously have to be carried out in the future as the code develops, and is only necessary if the memory constraints of existing HPC machines persist. For clarity, equation 2.13 shows the proposed multigrid preconditioner scheme for the within group second-order even-parity transport equation currently being built within SN2ND where PETSc is assumed to provide the solution at the final step (via algebraic multigrid or parallel SOR).

$$\begin{aligned}
 A_g \tilde{\psi}_g^+ = b_g^+ & & Y = M_g^{-1} \{b_g^+ - A_g \tilde{\psi}_g^+\} \rightarrow Y = M_g^{-1} X & & \text{block diagonal in angle} \\
 Y = M_g^{-1} X & & Y' = M_g'^{-1} \{X - M_g Y\} \rightarrow Y' = M_g'^{-1} X' & & \text{LT-multigrid} \\
 Y' = M_g'^{-1} X' & & Y'' = M_g''^{-1} \{X' - M_g' Y'\} \rightarrow Y'' = M_g''^{-1} X'' & & \text{h-multigrid} \\
 Y'' = M_g''^{-1} X'' & & Y''' = M_g'''^{-1} \{X'' - M_g'' Y''\} \rightarrow Y''' = M_g'''^{-1} X''' & & \text{algebraic multigrid}
 \end{aligned} \tag{2.13}$$

2.3 Implementation Issues with Parallelization in Energy

It is very important to note that these changes are being implemented into SN2ND in the most generic fashion possible, but that it does create a significant amount of coding momentum that will be difficult to remove in the future if deemed unnecessary or rewrite/replace if found to be inefficient. This of course is the Achilles heel of multigrid schemes since generic methodologies always ignore the underlying memory constraints of typical production machines. To counter this, we invested considerable time in FY2010 rebuilding the mesh data structures of UNIC which is more of an implementation issue and thus beyond the scope of this report (i.e. performance changes are insignificant). The general idea in the new version of SN2ND is to create management data structures usable by all levels of the preconditioner which are flexible enough to store most multigrid concepts (basically mapping arrays and interpolation methods) even though we will only be implementing a specific scheme in SN2ND at this point. The real advantage of course is that this scheme can be utilized directly in the work to develop and implement parallelization by energy.

Parallelization in energy begins by assembling equation 2.7 in terms of a single vector quantity in energy:

$$\begin{bmatrix} A_g & -B_g^T W_{g \rightarrow g}^- \\ -B_g & I - W_{g \rightarrow g}^- \end{bmatrix} \begin{bmatrix} \tilde{\psi}_g^+ \\ \tilde{\psi}_g^- \end{bmatrix} = \sum_{g' \neq g} \begin{bmatrix} W_{g' \rightarrow g}^+ & B_g^T W_{g' \rightarrow g}^- \\ 0 & W_{g' \rightarrow g}^- \end{bmatrix} \begin{bmatrix} \tilde{\psi}_{g'}^+ \\ \tilde{\psi}_{g'}^- \end{bmatrix} + \begin{bmatrix} q_g^+ \\ B_g^T q_g^- \end{bmatrix}. \quad (2.14)$$

Note that the left hand side of equation 2.14 was inverted using a source iteration scheme in equation 2.8. While this is efficient when using GS in energy, applying GMRES over the entire space-angle-energy system requires that we be able to define the operations:

$$A\Psi = b \quad \& \quad Y = M^{-1} \{b - A\Psi\} = M^{-1} X \quad (2.15)$$

As a consequence, we have to re-factor the existing operations in SN2ND such that we can perform the A matrix-vector operation in equation 2.15. To do this we define the compact notation for equation 2.14 as

$$H_g \tilde{\psi}_g = \sum_{g' \neq g} W_{g' \rightarrow g} \tilde{\psi}_{g'} + q_g, \quad (2.16)$$

and assemble it over all energy groups to obtain a form similar to equation 2.15.

While it would seem that these operations are already part of the GS iterative scheme and thus usable in a GMRES algorithm, they are not. The primary reason is because of the ownership rules dictated by the parallelization in space. The problems, and thus changes, arise from the fact that the even-parity vector in equation 2.14 is assembled over all elements in the mesh such that we get one spatial degree of freedom per mesh vertex, but the odd-parity equation is discontinuous over each element and thus we get a single degree of freedom per vertex of every element in the mesh. Close inspection shows that the B matrices in equation 2.14 are the primary trouble spot since they transfer information from the discontinuous odd-parity space to the continuous even-parity space (transpose works in the reverse direction) and thus ghosted information must be updated on the discontinuous space when applying B_g^T and on the continuous space when applying B_g . This means that the $W_{g' \rightarrow g}$ operations must also be applied on the larger ghosted space. This requires additional communication which doesn't appear in the GS scheme since the ownership of the odd-parity system is fully definable given the updated even-parity solution. Many mistakes were made during the rewrite of SN2ND associated with the ownership rules which are the fundamental reason for the delay in deploying the new version of SN2ND. The fact that we are trying to implement a new multigrid preconditioner scheme at the same time only complicated matters further. Unfortunately those mistakes will make it necessary to further modify the newly written coding to actually allow it to utilize parallelization in energy.

Regardless, with the ability to apply the coefficient matrix vector operation we can discuss the preconditioner we must apply in equation 2.15. Unlike the GS in energy solution scheme, the preconditioner over the space-angle-energy system does not require all levels of the preconditioner described by equation 2.13. Conceptually we anticipate using a block Jacobi preconditioner in energy (i.e. GS on the assigned energy domain), possibly combined with a multigrid scheme in energy to properly capture the energy redistribution. In this situation we will not need to apply the exact coefficient matrix-vector when solving the locally owned within-group systems and should instead be able to jump directly to the LT -multigrid preconditioner level in equation 2.13. This has substantial consequences in computational

effort compared with the existing GS algorithm assuming the additional communication required by the coefficient matrix-vector operation can be managed. While it is speculation that this will be effective, it does pose a problem for development since some of the existing time spent building a working GS in energy algorithm might not be used in the alternative GMRES algorithm.

3 MOCFE Development

The method of characteristics (MOC) [12-13] has been widely used in the past for two-dimensional reactor fuel assembly calculations due to its geometrical flexibility and its high degree of space-angle accuracy. The initial version created in FY2007 tested the MOC application on unstructured 3-D meshes while work in FY2008 focused on development of a version with some basic aspects of parallelism, addition of two-dimensional modeling, and acceleration of the within group source iteration algorithm with synthetic diffusion. The MOC methodology is typically termed a “matrix free” method because it does not require the storage of large matrices for each energy group. In reality, the formulation is not truly matrix free because it requires substantial memory resources to store the trajectory data used during the solve process. Thus, to solve larger than single assembly problems, it requires substantial memory resources to store the trajectory data along with significant computational resources to apply the numerical matrix inversions; hence parallelism can be quite useful.

As a consequence, the FY2009 work was focused on rebuilding the MOCFE solver such that memory and communication scaling was imposed, which required that the conventional source iteration scheme used to solve the within group equation be replaced with a GMRES algorithm [9,14]. Because we chose to incorporate parallelization in energy in addition to a new parallelization algorithm in space and angle and treatment of all element types in UNIC, the MOCFE solver was not fully finished in FY2009. The bulk of FY2010 work was focused on continuing the development of the parallel MOCFE capability where substantial difficulties were encountered with the back projection methodology and ray tracing issues associated with numerical round off.

3.1 Review of MOCFE Solution Scheme

The derivation for the method of characteristics starts with the within group equation, reiterated here as equation 3.1.

$$\hat{\Omega} \cdot \vec{\nabla} \psi_g(\vec{r}, \hat{\Omega}) + \Sigma_{t,g}(\vec{r}) \psi_g(\vec{r}, \hat{\Omega}) = W_g(\vec{r}, \hat{\Omega}) + S_g(\vec{r}, \hat{\Omega}) \quad g = 1, \dots, G \quad (3.1)$$

$$W_g(\vec{r}, \hat{\Omega}) = \int \Sigma_{s,g \rightarrow g}(\vec{r}, \hat{\Omega} \cdot \hat{\Omega}') \psi_g(\vec{r}, \hat{\Omega}') d\hat{\Omega}'$$

The first step in the MOC derivation is to introduce a new coordinate system which allows the $\hat{\Omega} \cdot \vec{\nabla}$ operator in equation 3.1 to be rewritten as a first-order mono-dimensional derivative. To do this, we project the incident portion ($\hat{\Omega} \cdot \hat{n} < 0$ where \hat{n} is the outward normal from the domain surface) of the problem domain boundary \mathbf{A} for a given direction $\hat{\Omega}$ to a plane A_{\perp} that is both exterior to the problem domain and perpendicular to the direction $\hat{\Omega}$ as shown in Figure 3.1.

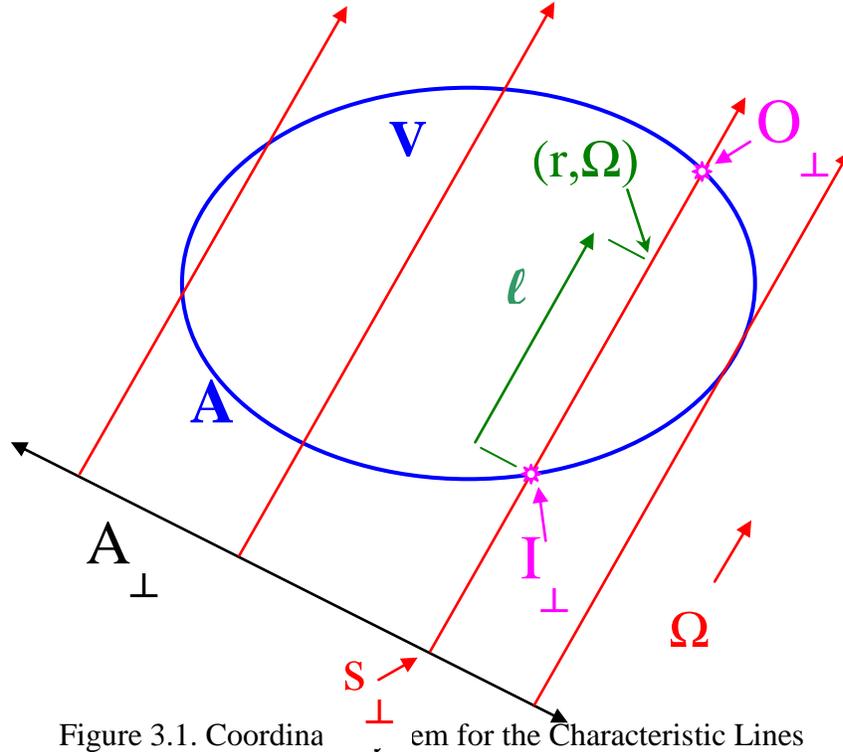


Figure 3.1. Coordinate system for the Characteristic Lines

We can define \vec{s}_\perp to be a two-dimensional coordinate on the surface A_\perp which intersects the problem domain boundary at points I_\perp and O_\perp . Between these points we can define the position ℓ to be the distance measured from the incident point I_\perp to some position $\vec{r} \in V$ within the volume. If the problem domain is convex, then the position $(I_\perp, \ell, \hat{\Omega})$ defines an alternative coordinate system to that used in equation 3.1. In this new coordinate system, the $\hat{\Omega} \cdot \vec{\nabla}$ operator becomes a partial derivative with respect to ℓ and we can rewrite equation 3.1 as

$$\left(\frac{\partial}{\partial \ell} + \Sigma_{i,g}(I_\perp, \ell) \right) \varphi_g(I_\perp, \ell, \hat{\Omega}) = W_g(I_\perp, \ell, \hat{\Omega}) + S_g(I_\perp, \ell, \hat{\Omega}), \quad (3.2)$$

where $\varphi_g(I_\perp, \ell, \hat{\Omega})$ is used to indicate that the flux vector is in the alternative coordinate system. As can be seen, this equation is dependent upon straight line paths (termed trajectories) that penetrate the problem domain in the direction $\hat{\Omega}$ which are referred to as the mathematical characteristics of the neutron transport equation.

The true vector unknown for MOC is the flux solution at each trajectory intersection point or the trajectory intersection flux. The storage of this vector is impractical, and for steady-state and adjoint calculations, unnecessary. Instead, the solution space is always recast to store a monomial expansion of the flux local to each element $(1, x, y, z, \dots)$ which is easily an order of magnitude or more smaller than the trajectory intersection flux. In previous work [2], an in depth derivation of how this is done was given which produced the equation

$$(I - \Upsilon L^{-1} \aleph \Lambda) \bar{\tau} = \Upsilon L^{-1} \bar{s}, \quad (3.3)$$

where $\bar{\tau}$ is the element-wise S_N flux solution (expanded in the monomial basis) and the matrix definitions are beyond the scope of this work.

The selected parallel algorithm incorporates a modified domain decomposition strategy to guarantee load balanced work and communication. With regard to equation 3.3, the outgoing trajectory flux solution is added to the vector space on each spatial subdomain thereby yielding the following governing equation

$$\begin{bmatrix} 0 & 0 & 0 \\ \Upsilon \tilde{L}_p^{-1} (\tilde{L}_p^{in} - (\aleph \Lambda \Upsilon)_p^{in}) & I - \Upsilon \tilde{L}_p^{-1} (\aleph \Lambda)_p & 0 \\ -\tilde{L}_p^{out} \tilde{L}_p^{-1} (\tilde{L}_p^{in} - (\aleph \Lambda \Upsilon)_p^{in}) & \tilde{L}_p^{out} \tilde{L}_p^{-1} (\aleph \Lambda)_p - (\aleph \Lambda)_p^{out} & I - \tilde{U}_p \end{bmatrix} \begin{bmatrix} \varphi^{in} \\ \tau \\ \varphi^{out} \end{bmatrix} = \begin{bmatrix} 0 \\ \Upsilon \tilde{L}_p^{-1} s \\ s^{out} - \tilde{L}_p^{out} \tilde{L}_p^{-1} s \end{bmatrix}. \quad (3.4)$$

In this equation, φ^{in} is a boundary condition obtained from the adjacent processor and the GMRES vector space only includes the element averaged flux solution components τ and the owned outgoing trajectory flux φ^{out} which we refer to as the trajectory segment flux. The definitions of the remaining matrices can be found elsewhere [2].

3.2 Problems Encountered in Ray Tracing

Numerous problems arose in FY2010 with the parallel ray tracing algorithm. After discussions with other researches also studying parallel MOC with domain decomposition, there appear to be as many approaches to obviating the problems as there are researchers on parallel MOC. To understand the problem we show a spatially decomposed VHTR assembly in Figure 3.2 where the upper three pictures show the material composition (left), decomposed domain (center) with owned elements colored and ghosted elements uncolored, and the visible piece of the mesh for all regions (right). The lower picture shows how the trajectories are broken as they cross the domain where it is obvious that several trajectories are reentrant on each subdomain.

The ray tracing methodology in MOCFE focuses on intersecting the elements in the domain by following the trajectories as they pass through each processor's piece of the mesh. This allows the storage pattern to follow directly with the way it will be used during the sweeping process of solving the system of equations. The alternative ray tracing methodology is to check all elements in the mesh for intersections with all trajectories for all angles. While the number of trajectories can be somewhat restricted by defining a bounding box for the local piece of the mesh on the back projection plane or even a bounding box for each element in the mesh, it still requires a considerably larger number of check operations than the implemented ray tracing scheme and it also requires that the ray tracing data be post processed to sequence it on each process. One advantage of taking the alternative approach is that we do not suffer the round off error associated with the ray tracing approach implemented in MOCFE where the trajectory path is not perfectly straight.

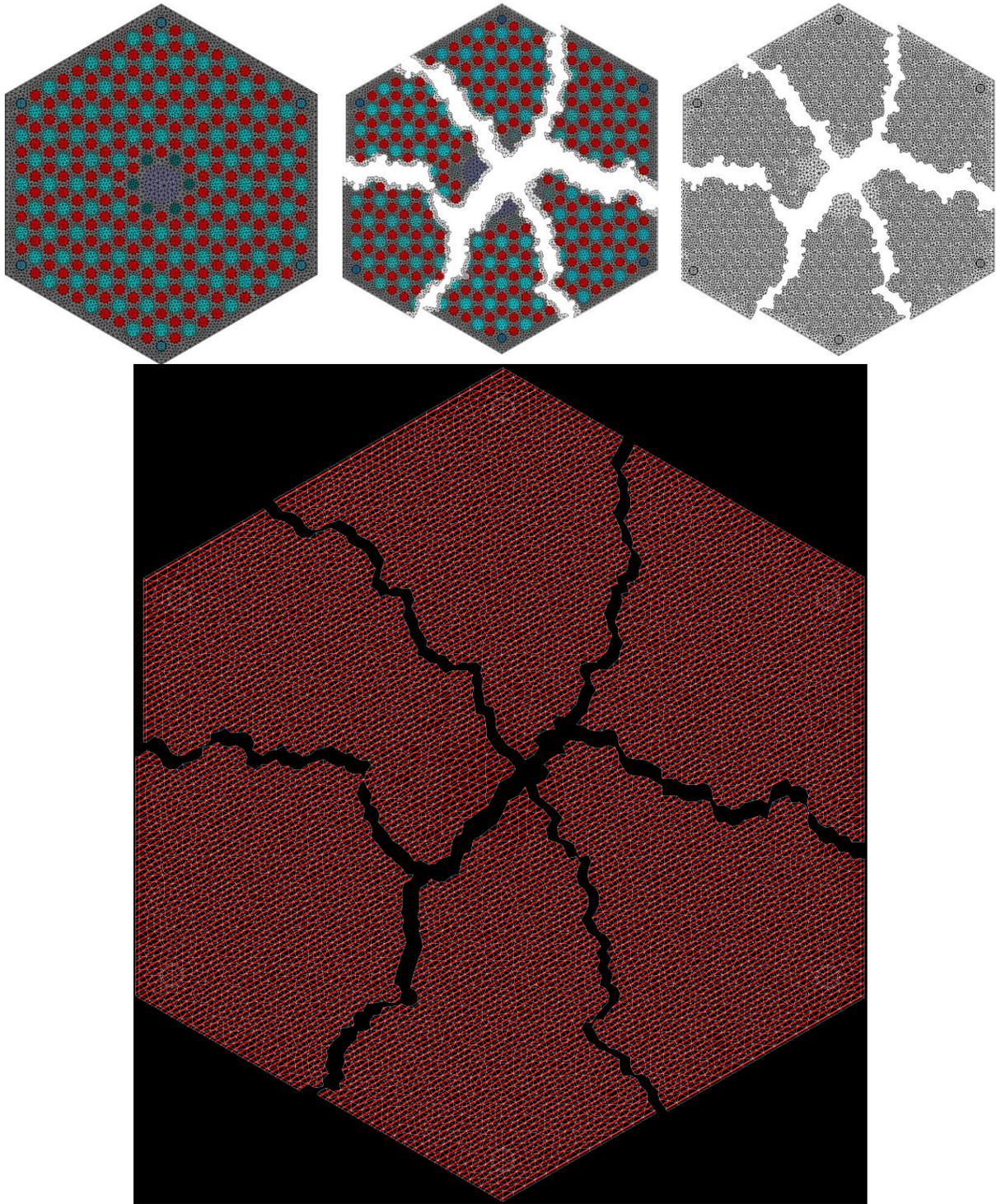


Figure 3.2. Example Decomposition of the Mesh and Segmentation of Crossing Trajectories

One issue we have observed with the ray tracing approach in MOCFE is when two adjoining processors that own adjacent problem domain surfaces end up calculating ray tracing data, and both processors state that the outgoing trajectory point on the domain is un-owned by either process or owned by both. To understand this, we show Figure 3.3 where a trajectory exactly intersects a vertex separating two adjoining processors.

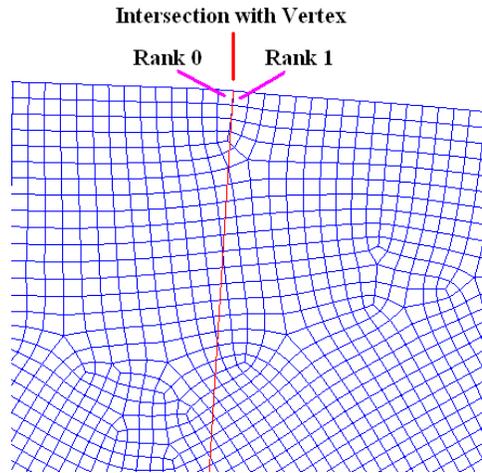


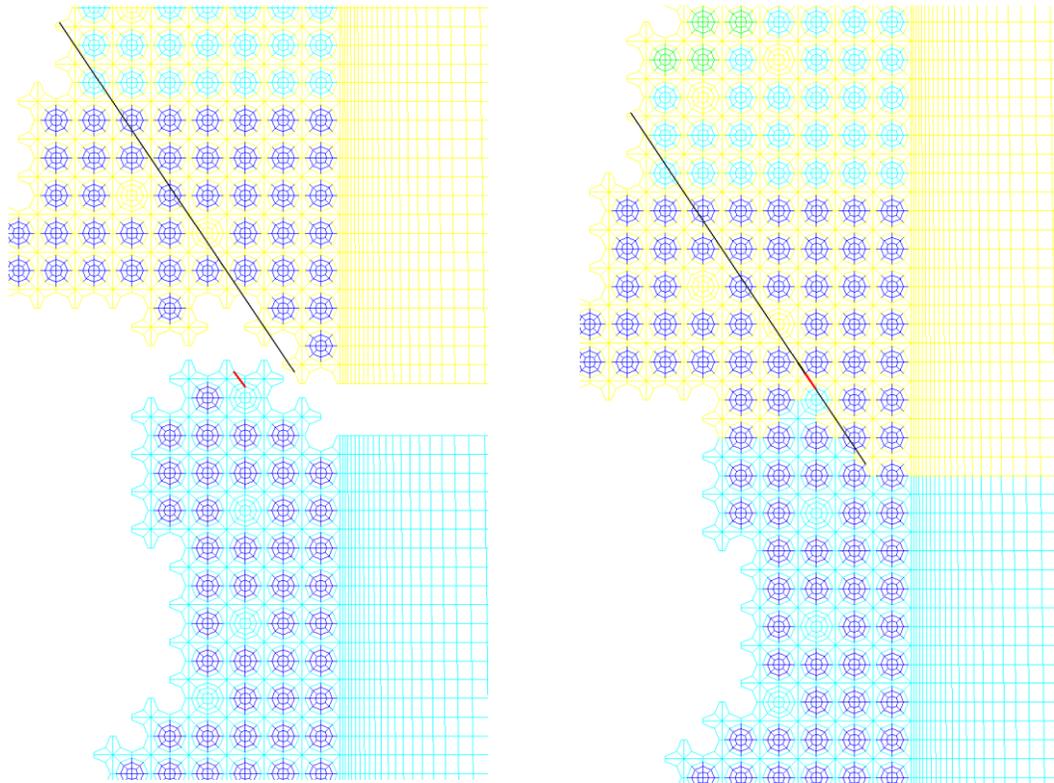
Figure 3.3. Example MOCFE Ray Tracing Problem on the Domain Surface

Because of round off error, the ray tracing through each processors subdomain has occasionally yielded the stated problem which is obviously unacceptable in either circumstance. One way to resolve this is to ensure that the trajectory is never defined in the first place which is accomplished by ensuring that the incident and exiting problem domain surfaces are preserved during the back projection algorithm. In two-dimensions this is quite trivial, but in three-dimensions, it is very difficult to devise a memory and computationally efficient scheme. The actual solution devised for MOCFE is to detect when this occurs and completely remove the trajectory from the ray tracing data redistributing its area to a geometrically adjacent valid trajectory (i.e. preserve neutron balance). As can be inferred, the probability of this type of intersection occurring with a domain decomposed mesh is relatively small.

Another very similar problem with the ray tracing algorithm has to do with ray casting and reentrant trajectories. To understand this issue we provide Figure 3.4 which shows part of a single trajectory as it crosses a parallel decomposed mesh.

In Figure 3.4, two adjacent pieces of a mesh are separated in the left hand picture and overlapped in the right hand picture. A single trajectory path is plotted as it crosses part of the mesh where we use **black** to indicate the total trajectory path and **red** to show a problem part of the intersection along the path. Similar to the problem described by Figure 3.3, round off error in the ray casting process causes slight differences on where each trajectory intersects the mesh and we find that the entry and exit point with respect to element can easily change. In Figure 3.4, the problem occurs because the red colored segment exactly hits a vertex and the ray tracing data in the lower mesh partition intersects an element in the owned portion (i.e. below the intersected vertex) while the ray tracing data in the upper mesh indicates that the trajectory fully goes through its owned portion of the domain (above the vertex). This causes

a vector ordering problem because one process states that the trajectory is reentrant and the other does not and chaos ensues. While the intersection distance is trivial (10^{-9} cm) in this case, there can be more severe instances which lead to the general question: which part of the ray tracing data is valid and invalid?



Separated Mesh for Adjacent Processors Contiguous Mesh for Adjacent Processors

Figure 3.4. Example MOCFE Ray Casting Problem on the Domain Interior

While this is easy to identify in the Figure 3.4 example, it is not necessarily clear when there are multiple valid paths through the domain. This can and has happened in meshes which contain sub-geometries such as assemblies with structured meshing where the parallel ownership is broken down one of the “structured grid lines” that create the structured mesh. Multiple processors can own pieces of the mesh along the grid line and a trajectory can (and typically does) travel in the same direction as the grid line and thus along a grid line. Because trajectory ownership across multiple processors is unknown at ray casting, the trajectory is allowed to start on all processors that are adjacent to the grid line thereby potentially creating multiple valid paths through the domain.

Combining these ray tracing issues with the fact that nuclear engineers routinely treat non-convex domains, makes all simple algorithms capable of resolving the ray tracing issues fail. To remedy the situation, a rather complex sequencing algorithm was implemented in MOCFE which requires detailed explanation. Figure 3.5 shows some potential ray tracing possibilities

represented in a one-dimensional space where the vertical lines represent mesh boundaries between adjacent parallel processors and the arrows represent trajectories crossing the domain.

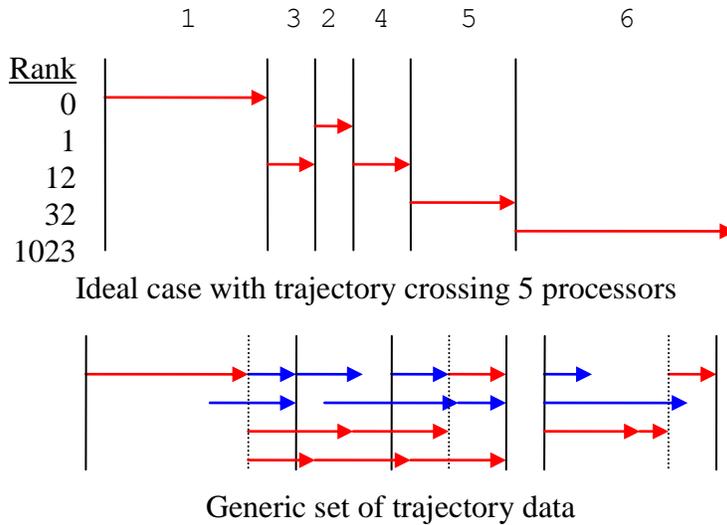


Figure 3.5 Example Segmented Ray Tracing Data

The upper example in Figure 3.5 shows the ideal case where there are no problems; the trajectories start and end exactly at the processor domain boundaries and the selection process for finding a path through the domain is clear (0-12-1-12-32-1023). The example at the bottom of Figure 3.5 shows a hypothetical mess of trajectory segment data where the trajectory is also reentrant on the global domain (the empty gap). The trajectory segments highlighted in red produce a “valid” path through the domain when combined and one can easily see the difficulty involved in determining the “best” path through the domain.

With the parallel algorithm chosen for MOCFE, we must ensure that all processors observe the exact same number and ordering of the segments along each global trajectory. This is independent of the fact that the global trajectory actually intersects the local subdomain since the trajectory segment flux is part of the global solution vector space in MOCFE. The algorithm created to solve this problem focuses on casting the above system into a coefficient matrix and rearranging rows and columns to form a lower triangular, non-singular matrix. To demonstrate it, the ray tracing data at the top of Figure 3.5 is used which starts by numbering the segments 1 through 6 as shown at the top of Figure 3.5. This ordering corresponds to the rank ordering of the reported trajectory segments obtained using the mpi “scan” function.

With this setup, we can define the forward adjacency for each trajectory segment (row) with the connecting trajectory segment (column) that immediately follows it. Placing a one at this row/column connection is sufficient to define the matrix system which we find for the simple example to be:

$$\begin{array}{c}
 2 \ 3 \ 4 \ 5 \ 6 \\
 1 \ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \\
 2 \ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix} \\
 3 \ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 4 \ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\
 5 \ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}
 \end{array} \tag{3.5}$$

First we note that this matrix is not singular and that we have removed the row corresponding to segment 6 and the column for segment 1 since neither is definable in this context. If we apply the same approach to the example at the bottom of Figure 3.5, we would get a singular system because there is no possible connection about the reentrant point (gap).

Continuing, the next step in the algorithm is to simultaneously reorder the rows and columns such that we have an identity system, the first step of which swaps the first and second columns to make the first row lower triangular.

$$\begin{array}{c}
 3 \ 2 \ 4 \ 5 \ 6 \\
 1 \ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 2 \ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix} \\
 3 \ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \\
 4 \ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\
 5 \ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}
 \end{array} \tag{3.6}$$

The next step requires us to swap the first and second rows to indicate that the 3rd trajectory segment follows the first as specified by the first non-zero column in the first row

$$\begin{array}{c}
 3 \ 2 \ 4 \ 5 \ 6 \\
 1 \ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 3 \ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \\
 2 \ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix} \\
 4 \ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\
 5 \ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}
 \end{array} \tag{3.7}$$

Given that there are no non-zeros in the upper or lower triangular portion, we have successfully constructed a valid path through the domain with the set of segments: 1,3,2,4,5,6, which is the correct solution.

If we assume there is a small trajectory (10^{-9} cm) segment appearing after segment 4 in the upper example in Figure 3.5 we get the following system and its final reduced form

$$\begin{array}{c}
 \begin{array}{cccccc}
 & 2 & 3 & 4 & 5 & 6 & 7 \\
 1 & \left[\begin{array}{cccccc}
 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 1 & 0 & 0 & 0 \\
 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 1
 \end{array} \right] \\
 2 \\
 3 \\
 4 \\
 5 \\
 6
 \end{array}
 \rightarrow
 \begin{array}{c}
 \begin{array}{cccccc}
 & 2 & 3 & 4 & 6 & 7 \\
 1 & \left[\begin{array}{cccccc}
 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 1 & 0 & 0 \\
 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1
 \end{array} \right] \\
 2 \\
 3 \\
 4 \\
 6
 \end{array}
 \rightarrow
 \begin{array}{c}
 \begin{array}{cccccc}
 & 3 & 2 & 4 & 6 & 7 \\
 1 & \left[\begin{array}{cccccc}
 1 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1
 \end{array} \right] \\
 3 \\
 2 \\
 4 \\
 6
 \end{array}
 \end{array}
 \end{array}
 \quad (3.8)$$

This small trajectory appears as a connection to trajectory 4, but since it is not long enough to connect to trajectory 7 (or 6), it is registered in equation 3.8 as a completely zeroed row. In this case, the algorithm removes these rows (and the corresponding columns) which corresponds to the fact that we should remove the small trajectory lengths from the ray tracing data and get the valid set of segments: 1,3,2,4,6,7. Note that this algorithm inherently depends upon the fact that there is only a single identifiable outgoing trajectory point.

In the more complicated example at the bottom of Figure 3, the reentrant position will yield a systemic truncation of the adjacency matrix where only the valid part of the path past the reentrant point is left. Fundamentally this system fails because it indicates that row 1 must be removed which is invalid. In this situation we return the first path location which is not invalid and a forced “trajectory jump” is setup to define an adjacency link across the gap. The trajectory segment selected for the jump is selected based upon its proximity to the known valid segment in addition to its length (longest is desired). If there are more than one reentrant crossing for a given trajectory, the code will continue to add jumps until a valid path is found from the exiting trajectory (the one identified to have the highest exiting point along the trajectory path) and the entering trajectory (the one with the lowest exiting point). This does not mean that the code will link the flux across the gaps, because it inherently understands that the domain is not convex because of the number of domain boundary surfaces crossed by each trajectory.

Because of the complicated nature of this algorithm, a separate testing routine was built into MOCFE such that it was validated with numerous hypothetical trajectory segment adjacencies. While this algorithm can be implemented in a scalable way, it presently is implemented in serial. This means that all processors duplicate the effort of sorting the trajectory data although the amount of information being simultaneously processed is controlled to avoid memory related limitations and maximum efficiency. At present the time spent in this algorithm on calculations using 1000 processors is inconsequential. As a final note, round off error during the ray tracing procedure can cause the segment end points to be misaligned above the limits of any arbitrarily small tolerance. Consequently, MOCFE almost always reports trajectory “jumps” even though the lengths are rather small $\ll 10^{-7}$ cm. As an example, a two-dimensional model of a full core VHTR with 36 ray tracing directions and 1.8 million trajectories with a 32 processor spatial decomposition reported 2,344 trajectory jumps all having trajectory distances well within the expected tolerance settings.

Most of these ray tracing issues can be resolved if a full back projection of all elements in the domain is carried out. While this is possible on small geometries such as single assemblies, for full core problems this can be a considerable burden and thus unwise since it can increase the number of trajectories by over an order of magnitude and thus decrease the average

trajectory area well below the typical 0.01 cm^2 used in most MOC calculations. While these problems are not unique to MOCFE, some developers [15] pursued an alternative methodology which obviates the issues entirely.

One approach is to use a modified domain decomposition rule where a structured grid is overlaid on the unstructured grid. The unstructured mesh is geometrically split (not vector split though) along each surface of the overlaid grid and all elements within the structured grid boxes are assigned to individual processors. The trajectory paths through the domain are trivial to construct with this scheme since it eliminates reentrant trajectory lines and a simple ownership rule can be defined for any trajectories that intersect the edges or corners of the structured grid. The only question is how this approach achieves a good load balance since the number of elements per box will not be equal for most reactor problems and the number of trajectories intersecting each box will not be equal. The entire purpose of the mesh partitioning algorithm in MeTiS, which produces non-intuitive mesh boundaries as seen in Figure 3.2, is to algorithmically determine the mesh partitioning such that the local work is balanced and communication is minimized.

3.3 A Scalable Back Projection Algorithm

As will be discussed later in this section, the initial scaling studies of the VHTR benchmark demonstrated that the serial back projection algorithm in the two-dimensional MOCFE solver failed quite dramatically and prevented the scalability numbers from being meaningful. The fundamental reason for the failure was the unexpected amount of memory required to store the back projection data, over a million back projection (BP) surfaces and $> 1,000,000$ trajectories per direction, which overwhelmed the 512 MB limit currently placed on each processor of BlueGene/P. This was of course a combination of the attempt to use a full back projection and an angular cubature which was not coherent with the geometry of the VHTR (i.e. a 60 degree rotational basis). This was not obviously expected (hence the serial algorithm) and time was invested in FY2010 to create a parallel algorithm for not only the two-dimensional option, but also the three-dimensional option of MOCFE.

The purpose of the back projection (BP) algorithm is to define the set of starting points for all of the global trajectories traversing the problem domain. We can separate this into the following distinct operations:

- 1) Project the local domain surfaces to the BP plane
- 2) Identify the portion of the BP plane owned by each process
- 3) Redistribute the locally generated surfaces to the process that they were assigned
- 4) Mesh the locally owned portion of the BP plane
- 5) Assign trajectory starting points within every element on the BP plane
- 6) Redistribute the trajectories from the BP plane to the intersected processors

The key to efficiency in the new algorithm is how to perform operation 2 such that operations 3-5 are all scalable with respect to memory and communication. The Frameworks component of SHARP was initially tasked with devising a scalable method for the 3-D MOC back projection. After some time, it appears as though a practical implementation using the frameworks tools will require so much development time that the deployment of a 3-D MOC

capability would be delayed by several years. To avoid this delay, a simplified approach was implemented which is shown graphically in Figure 3.6.

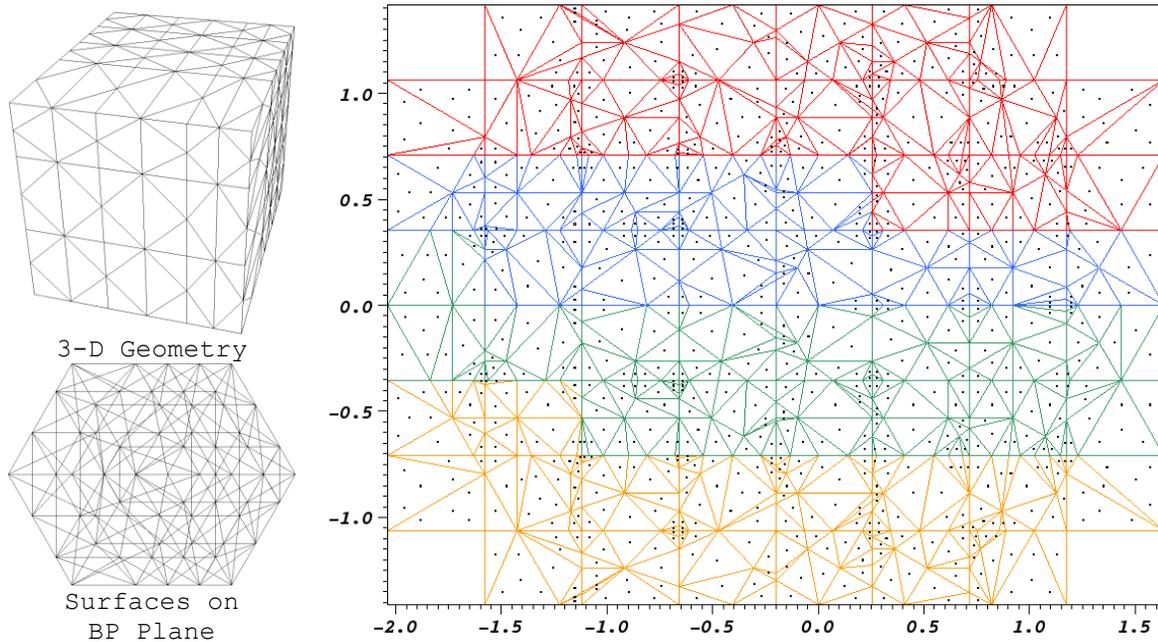


Figure 3.6 Example Four-Processor Back Projection

From Figure 3.6, the surfaces from each element in the 3-D mesh of hexahedral elements (upper left) are projected to the BP plane generating a large number of overlapping triangles (bottom left). In 2-D, the BP plane is a line and the 2-D elements create a set of overlapping line segments on that line. In both 2-D and 3-D, the boundaries of the projected domain are easy to identify; smallest and largest position on the line for 2-D or a square surrounding all of the BP surfaces for 3-D as seen in Figure 3.6.

Focusing on the 3-D algorithm, maintaining all of the lines of the BP surfaces is the difficult task being studied which is expected to take several years to develop. To circumvent the issue, the current algorithm assumes that a mesh constructed from the vertices generated by the BP surfaces is sufficient to generate the necessary trajectory points. Given the bounding box, we can partition it into smaller boxes, termed subboxes, which is a multiple of the number of processors in the domain. In Figure 3.6, we specified four-processor spatial domain decomposition and created 64 subboxes within the bounding box (8x8). Since the subbox that each BP point lies within can be trivially determined, we only need to determine how many subboxes to assign to each processor such that each processor has the desired average necessary to guarantee load balancing in meshing and communication of the BP vertices.

The coloring of the triangles in Figure 3.6 shows the rank assignment of each subbox for this particular problem. The triangles are generated using the Triangle open source package [16] which has proven to be quite efficient meshing problems with upwards of 1,000,000 vertices. Because Triangle assumes a convex domain we also include the four corner points of each subbox and use Triangle to individually mesh each subbox where any points that lie on the edges of the subboxes are duplicated on adjacent subboxes to create a conformal mesh.

Note how the total domain projection seen in the bottom left picture of Figure 3.6 is preserved in the triangulation result on the right, but not all of the interior triangle surfaces are preserved. This ensures conservation on the whole domain but not on any given problem domain surface and we can expect some amount of inaccuracy when we apply reflected boundary conditions. Of course this error is reduced as the mesh is refined which happens to be the same requirement already imposed for a reflected boundary condition given that we utilize a flat flux approximation in MOCFE.

The user input controls the number of trajectory points that will be generated from each triangle and, since the preceding algorithm is focused on load balancing the meshing and communication of BP vertices, we can safely say that it will not be perfectly scalable with regard to communicating the trajectory data itself. This should be acceptable since every processor still has to discriminate whether the received trajectory starting points intersect the local domain which is done most efficiently using a bounding box around the local mesh. From there each processor is responsible for further checking whether the locally “nearly intersecting” trajectory data actually intersect the incident surfaces of the local mesh.

The 2-D algorithm is conceptually the same, but much simpler in many areas. First, it is easy to maintain all interfaces of the BP elements since they form a series of overlapping line segments. Second, rather than use a meshing package, we only need to sort the BP vertices produced by all sublimes and eliminate points which are identical (we assume a minimum point spacing and thus trajectory area of 10^{-6}). As a final note, from Figure 3.6 one can see that the overall problem domain boundary is not preserved in the meshed routine. This can lead to problems since the projected area is inconsistent with the actual domain projected area. The inclusion of the intersection points of the domain surfaces with the subboxes effectively resolves this issue although it is not shown here. Given the other work being performed in UNIC, we were not able to fully test the impact of these changes or the accuracy of the 3-D MOCFE back projection algorithm.

3.4 Automatic Optimization Adjustor Algorithm for MOCFE

Part of FY2010 was spent building an auto adjustor routine for the serial version of the MOCFE code. This is needed to not only guarantee convergence, but also to attempt to reduce the computational effort. To build an adjustor routine, the algorithm must monitor the current effort involved in each iterative system and estimate the impact of the targeted error settings on the spectral radius of the system in energy, the within group systems, and the within group preconditioner systems in addition to the dominance ratio of the power method. For the power method we measure the error criteria

$$\varepsilon_{fission} = \frac{\|Q^{i+1} - Q^i\|}{\|Q^{i+1}\|} \quad \varepsilon_{flux} = \frac{\|\Psi^{i+1} - \Psi^i\|}{\|\Psi^{i+1}\|} \quad \varepsilon_{eigenvalue} = \frac{k^{i+1} - k^i}{k^{i+1}}. \quad (3.9)$$

For the Gauss-Seidel algorithm in energy or the Krylov in energy we measure

$$\varepsilon_{\Psi, Krylov} = \frac{\|b - A\Psi^{j+1}\|}{\|b - A\Psi^1\|} \quad or \quad \varepsilon_{\Psi, GS} = \frac{\|\Psi^{j+1} - \Psi^j\|}{\|\Psi^2 - \Psi^1\|}. \quad (3.10)$$

For each within group system where Krylov or source iteration is used we define

$$\mathcal{E}_{\Psi(g),Krylov} = \frac{\|q_g - A_g \Psi_g^{k+1}\|}{\|q_g - A_g \Psi_g^1\|} \quad \text{or} \quad \mathcal{E}_{\Psi(g),SI} = \frac{\|\Psi_g^{k+1} - \Psi_g^k\|}{\|\Psi_g^2 - \Psi_g^1\|}. \quad (3.11)$$

Finally, for each within group preconditioner system we define

$$\mathcal{E}_{\phi(g),Krylov} = \frac{\|s_g - C_g \phi_g^{m+1}\|}{\|s_g - C_g \phi_g^1\|}. \quad (3.12)$$

At present, MOCFE only has an algorithm to control the GS case where a Krylov solver is used on the within group equation. The algorithm does not control the preconditioner level where we have set the error conservatively as $\mathcal{E}_{\phi(g),Krylov} = 0.01$ for non-domain decomposed problems and $\mathcal{E}_{\phi(g),Krylov} = 0.0075$ for spatial decomposed domains (parallel spatial applications). The set of error criteria defined by equation 3.9 are controlled by user input thus leaving the adaptation algorithm focused on controlling the error targets in equations 3.10 and 3.11. To develop the control algorithm, several reactor problems of interest were used which include an ABTR assembly (9 and 33 groups), a PWR assembly (23 groups), a VHTR assembly (23 groups), the C5G7 benchmark problem (7 groups), a CANDU assembly problem (23 groups), and a large “drawer homogenized” fast reactor problem (9 groups). In all of the problems angular cubatures were used ranging from S_4 to S_8 which are acceptable for the purpose of developing the adjutor.

The initial step of the algorithm is to determine what the expected “slope of convergence” is for each system on each calculation. The slope of convergence is defined by normalizing each error measurement with the initial measurement and using a \log_{10} conversion. With multiple values, a least squares linear fit can be applied to get the slope which is an effective measure of the dominance ratio. In MOCFE the slope of the fission source error defined in equation 3.9 is used along with the slope of the “flux norm” which is the numerator from equation 3.10.

To understand how this data is used, Figure 3.7 plots the cited error measures from equations 3.9 and 3.11 and the “flux norm” for the 33 group ABTR problem where no auto adjustment is applied. This calculation had input error targets of 5.0×10^{-7} , 5.0×10^{-6} , and 5.0×10^{-6} , for the eigenvalue, fission source, and flux error, respectively, and the GS error target was fixed at 0.25 and the source iteration (SI) error target for each group was fixed at 0.02. Note that the ABTR has no upscattering and that the ability to achieve a 0.25 error on GS is strictly determined by the error achieved in the within group systems. Using the data from Figure 3.7, the slope of the fission source (and flux error and eigenvalue error) can quite easily be found to be -0.64 if the results in the first two fission source iterations are ignored. Each within group system (WGS) flux error curve only achieves a slope near this after the sixth fission source iteration. Before the sixth fission source iteration we can clearly identify some of the lower energy groups (32 as an example) to have nearly flat slopes of -0.1. These lower energy groups do not contribute significantly to either the fission source or the flux error because the number of neutrons reaching this level is rather small (they are absorbed before they reach these energies since this problem is an infinite lattice of fuel assemblies), thus the rate of convergence of the fission source and eigenvalue are unaffected. The other key piece of information is the slope of the flux norm which is found to be -0.57 from Figure 3.7.

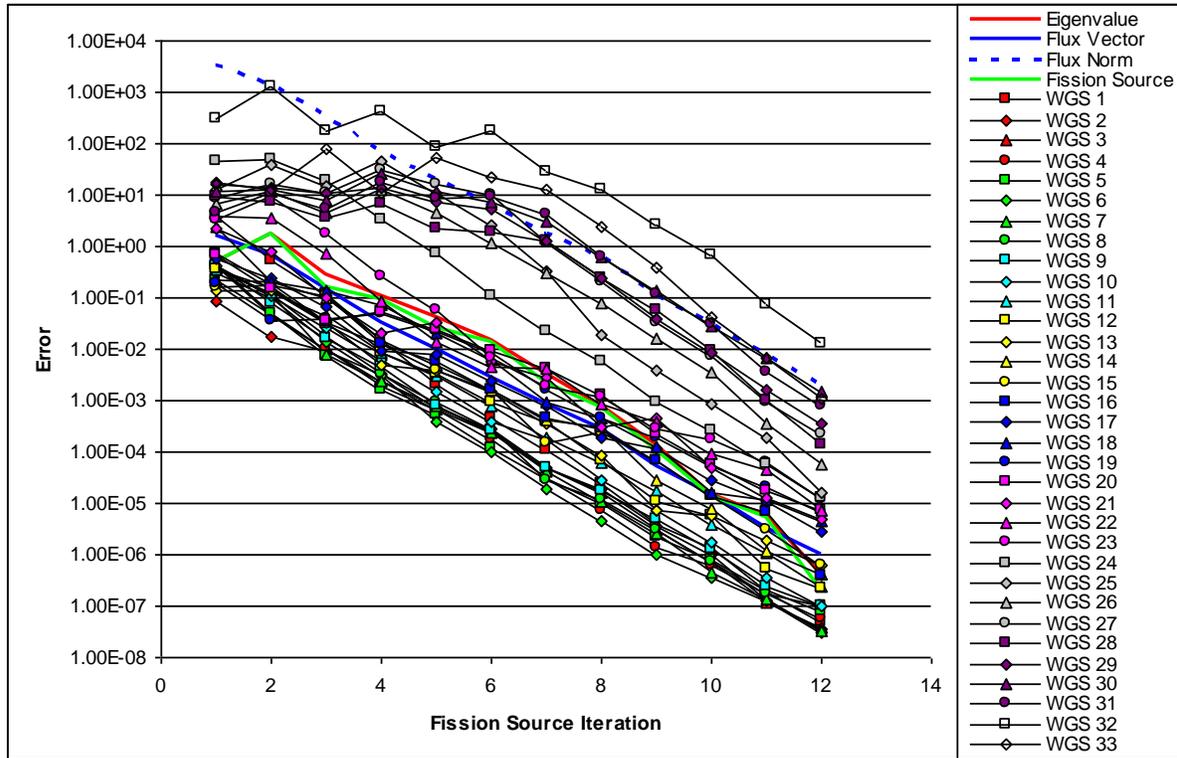


Figure 3.7 Example Error Measures for the 33 Group ABTR Test Problem

From these calculations it is clear that the exact inversion of the coefficient matrix during each power (fission source) iteration is unnecessary as evident by the fact that the flux norm slope in Figure 3.7 is much more negative (converging faster) than the slope derived from the fission source error (say -0.85 compared with -0.64). It is easy to understand how this happens because of the “relative improvement” error target which forces the MOCFE iterative solution scheme to always perform some amount of work before quitting. Thus 0.01 would infer that the error in the flux is reduced by two orders of magnitude while 0.001 would infer that the error is reduced by three orders of magnitude. Observations on eigenvalue problems to date have rarely indicated these error targets for either SN2ND or MOCFE should be less than 0.01.

After analyzing the results of the other selected benchmark problems it was determined that the selected fast reactor problems exhibited dominance ratios of 0.3 to 0.6 while the selected thermal reactor problems had dominance ratios of 0.65 to 0.90. Without Tchebychev acceleration, the dominance ratio was typically much higher. Figure 3.8 shows the convergence ratio of some hypothetical norm given the dominance ratio (DR). Given that a measurement of the dominance ratio can be obtained using the slope of the fission source error ($DR \approx 1 + \text{slope}$), the minimization of computational effort on each outer will yield a slope of the flux norm that is less than or equal to the slope of the fission source error. An initial slope target is set based upon the problem being executed (detect upscattering: -0.3 for fast systems and -0.65 for thermal systems). The relationship of the initial slope (computed at the fifth fission source iteration using the third, fourth, and fifth iteration results) to the ideal slope determines the first adjustment made in the targeted error criteria for GS or equation 3.10.

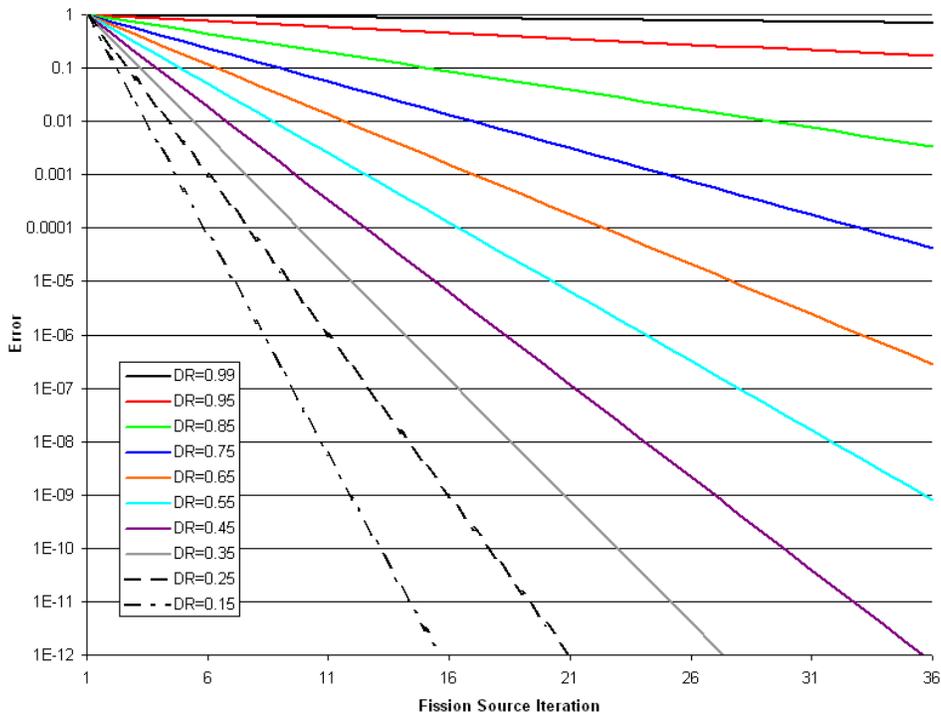


Figure 3.8 Example Dominance Ratio (DR) Impact on Solution Convergence

If an increase in the slope is observed over a few iterations, this indicates that the convergence target was relaxed too much and an appropriate change is made. If a decrease in the slope is observed then a decision is made depending upon the magnitude of the change as to whether the targeted error is tightened or loosened. In general the algorithm tries to obtain the maximum negative slope possible using the weakest error control possible on the GS system. This part of the algorithm should be identical to one used for the Krylov (GMRES) alternative.

The next part of the adjustor routine focused on redistributing the computational effort according to the importance of each within-group system (WGS) to the flux norm which can be considered “group balanced work.” To better understand what this means, the results from the VHTR benchmark problem shown in Figure 3.9 without group balancing can be contrasted with the same problem executed with group balancing in Figure 3.10 where neither figure displays the user targeted error criteria, but it was the same in both. In Figure 3.9, one can see that most of the within group systems report errors of between 10^{-5} and 10^{-8} at the end of the calculation. This is what happens when a flat target is applied to all within group system targets in equation 3.11 such as 0.02. In reality, some of the energy groups will not contribute a substantial amount of error to the fission source or the flux error, both of which are targets specified by the user as final exit criteria. As a consequence, those groups with error tracking at the bottom of Figure 3.9 can be considered “over converged” relative to those at the top. In Figure 3.10, the adjustor algorithm selectively weakens the error targets on some groups and strengthens it on others such that the error contribution is more evenly distributed which is observable when comparing the two figures as a reduction in the variance in the group wise reported errors.

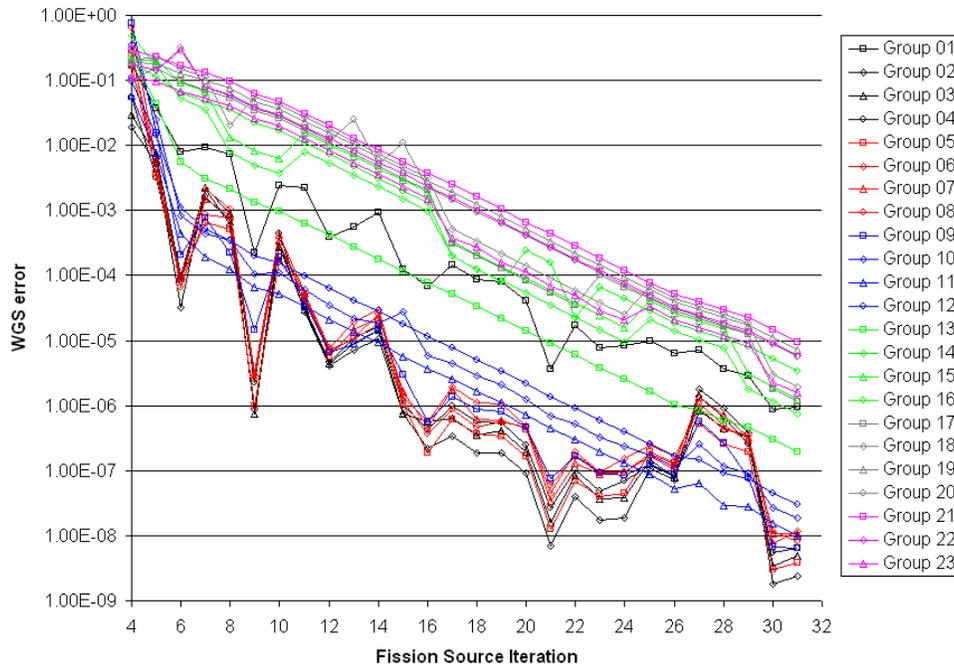


Figure 3.9 Within-group Results for the VHTR without Group Balancing

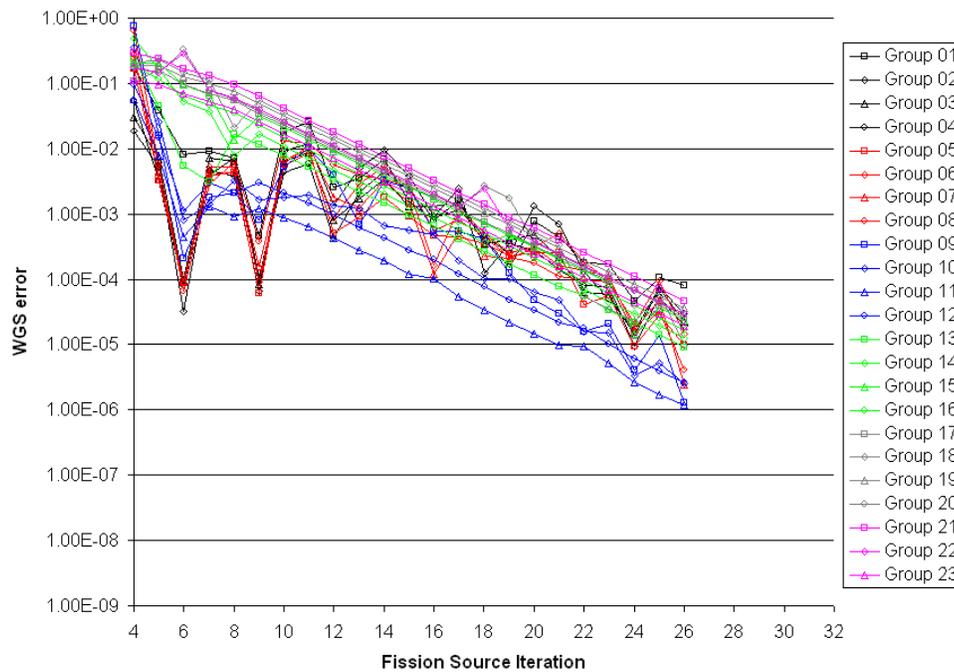


Figure 3.10 Within-group Results for the VHTR with Group Balancing

This part of the algorithm amounts to a considerable amount of computational effort, 20-30%, but there does not appear to be much more room for reductions using this technique (i.e. decreasing the variance further) because the error in the higher energy groups feeds into the lower energy groups via scattering. What is most important to note is that the overall rate of convergence of the flux solution is not altered using the algorithm, but, the fact that it

performed five fewer fission source iterations indicates that the approach did improve the rate of convergence of the fission source likely by remove spurious error contributions from the slowly converging systems.

The final part of the adjuster that was started but not implemented was adjustment of the preconditioner error target in equation 3.12. Of all of the error measures, this is the easiest to measure and adjust, but unfortunately it is the least important. Overall, the diffusion synthetic acceleration (DSA) operations with the fixed setting of 0.01 or 0.075 consume a maximum of 2-3% of the computational burden in the MOCFE solver for any of the cited benchmark problems. Thus erroneously weakening the preconditioner can be quite detrimental if it causes significant increases in the spectral radius of the within group system. Given that an abject failure was observed in one test calculation when the tolerance was switched from 0.01 to 0.05 (i.e. the preconditioner caused divergence), one would think any attempt to weaken the current DSA algorithm to be unwise. Regardless, given that the number of WGS iterations (either SI or Krylov) required to achieve the targeted error between two consecutive WGS solve operations is known and the number of preconditioner iterations during that process is known, the path to alter the preconditioner error target is relatively easy to find by comparing the ratios

$$ratio = \left[\begin{array}{c} \frac{\mathcal{E}_{DSA}^{j+1}}{\mathcal{E}_{DSA}^j} \\ \frac{Iterations_{WGS}^{j+1}}{Iterations_{WGS}^j} \\ \frac{Iterations_{DSA}^{j+1}}{Iterations_{DSA}^j} \end{array} \right]. \quad (3.13)$$

Given this information, the error target at the WGS level between the two consecutive is checked to ensure that it did not change and that the total number of Gauss-Seidel iterations in energy did not change (otherwise these numbers are partially dependent upon that change). If the DSA error target was relaxed previously and the number of WGS iterations (SI or Krylov) decreased then the change in the error was valid and further relaxation of the error is possible. If, however, the number of WGS iterations increased significantly than the algorithm reverts to the previous value immediately. Alternatively, if the DSA error target was tightened and the number of WGS iterations decreased, then the computational effort was reduced and further tightening of the tolerance can be considered. If the number of WGS iterations increased didn't change significantly then there is no reason to change the current setting, but the algorithm optionally chooses a value between the previous failed and successful setting. All of this assumes the application of the sweep operator is more expensive than DSA which is true for both MOCFE and SN2ND.

In summary, the preceding auto adjusting algorithm significantly reduces the computational effort consumed by conventional fixed target schemes. While the adjuster did perform excellently for the fast reactor problems and good for the high dominance ratio thermal reactor problems such as CANDU, some of the other problems, such as VHTR still came back with timing results near those obtained using various hardwired flat by group values indicating some degree of inefficiency still persists in the current search algorithm. Nevertheless, the current adjuster does provide the one key goal of this work: a reliable solver

which doesn't require substantial user involvement in defining or redefining convergence control parameters.

3.5 Initial Scalability Results for MOCFE

Part of the goal for FY2010 was to perform a detailed assessment of the parallel MOCFE capabilities. While this cannot be considered a final assay given the rather unclear path to a scalable algorithm, it should give some indication of what needs to be worked on to progress MOCFE to a scalable solver. To begin this work, the whole core VHTR problem using 2 groups was selected for study since it was sufficiently large (2,572,984 elements) to partition into more than 1024 pieces in space. The calculations were also restricted to running on BlueGene/P (which has processors over an order of magnitude slower than conventional serial machines) because of the ample time and short queue time available to run the test calculations. As expected, the initial scalability numbers shown in Table 3.1 are not good.

Table 3.1 Preliminary Strong Scaling in Space Numbers for MOCFE

Spatial Partitioning	Elements Per Process	Trajectories	Total DOFs (millions)	Total Time (sec)	Fission Source Iter.	WGS Krylov Iter.	Strong Scaling Space
64	40202	164895	56	2422	4	185	1.0
256	10050	234307	84	3153	8	329	0.69
512	5025	306297	123	3511	30	155	0.55
1024	2512	409079	197	4100	18	456	0.41
2048	1256	564453	344	2609	16	496	0.31
4096	628	784589	635	2186	18	515	0.22

Of course these numbers are rather convoluted since the small processor cases were not able to finish given the machine run time limits of BlueGene/P (< 512 node jobs are only allowed to run 60 minutes) and there are numerous issues associated with not being able to use a full back projection at all levels.

To begin, note that the number of elements assigned to each processor decreases consistently. This is in fact a terrible measure of scaling since, for MOCFE, the size of the element is a key to defining the local work, not the number of elements. This is primarily because the size of the element dictates the number of trajectories that will cross that element along with the number of intersections that will occur on the process which owns it. To display this viewpoint, Table 3.2 gives a breakdown of the vector size for the VHTR problem with no parallelization by angle. From Table 3.2 one can see that the per-processor number of elements is highly variable and, at the 4096 level, the RMS number indicates there are likely no processors with the average element count (strongly biased towards the minimum and maximum values). This distribution was obtained by weighting the MeTiS decomposition using the element volume in an attempt to balance the number of trajectories per process. A quick view of the space-angle degree of freedom (DOF) per process in Table 3.2 shows that the weighted partitioning did not have the desired effect since the DOFs are wildly varying. Such a load imbalance is not observed in SN2ND where a mere 1.05 to 1.10 difference in workload is generally assigned by MeTiS without using any weighting. The typical outcome of a load imbalance like this is to destroy the scalability as observed in Table 3.1.

Table 3.2 Variation in the Number of Elements and Trajectory Data

Spatial Partitioning	Elements Per Process				Space-angle DOF Per Process			
	Min	Max	Avg.	RMS	Min	Max	Avg.	RMS
64	11250	111415	40202	52977	63150	515632	435210	148444
256	5365	30298	10050	27559	18468	179042	164998	49229
512	2426	15239	5025	14437	13054	131254	119770	29280
1024	1052	7737	2512	7336	7550	103324	95978	18867
2048	433	3896	1256	3743	4926	76492	84057	13912
4096	199	1995	628	1895	3108	74818	77513	11446

The additional fact that the full back projection was not usable on BlueGene/P (memory) requires that the scalability numbers be adjusted to compensate for the changing number of trajectories also reported in Table 3.1. Part of the requirement of the parallel algorithm is to ensure that the incident elements of each subdomain are intersected properly by each trajectory which alters the back projection with each increase in the number of spatial subdomains. From Table 3.1, the increase in the number of trajectories is quite considerable and the computational time should go up quite dramatically as observed in Table 3.1. While the total time can easily be rebalanced using these numbers it doesn't properly account for the computational effort involved since the number of DOFs are also increased because of the trajectory segment flux. While a linear increase in the computational effort is not correct, it was used to be conservative and one should consider the actual scalability of MOCFE to be better than reported in Table 3.1.

The final issue that needs to be accounted for in Table 3.1 is the variable number of fission source iterations failure to reach convergence at the lowest level. This variation in fission source iteration is a consequence of the auto adjustor which is trying to minimize the amount of work. To factor this in, the scalability numbers are changed from total solution time (typical meaning) to time per Krylov iteration. This is a bad idea since the first three fission source iterations take a significantly different time (initial guess for Krylov solution is bad) than the last ten fission source iterations, but we are left with few choices. Combining all of these factors, it is not surprising to see the 22% scalability number and one should really consider it to be mostly unreliable.

What is useful from this exercise is information about the load imbalance on any given calculation. The large variance in the assigned space-angle DOF in Table 3.1 was observed to cause a load imbalance of 6-10 for the within group system (WGS) solve which dominates the total computational time for MOCFE. After some study, this was primarily identified to be a result of the number of trajectories intersecting each domain rather than an inherent problem with MOCFE. To understand why this is occurring, Figure 3.11 shows an 8, 16, 32, and 64 processor decomposition of the VHTR problem where the coloring only considers an 8 processor distribution. As can be inferred from the ray tracing problems earlier, the ray tracing data itself is strongly dependent upon the element density with respect to the direction of travel and thus some of the sub-domains will see strong variations in the number of crossing trajectories depending upon the direction of travel. Decomposing the problem with respect to angle will therefore cause a severe load imbalance which is why it was eliminated from Table 3.1.

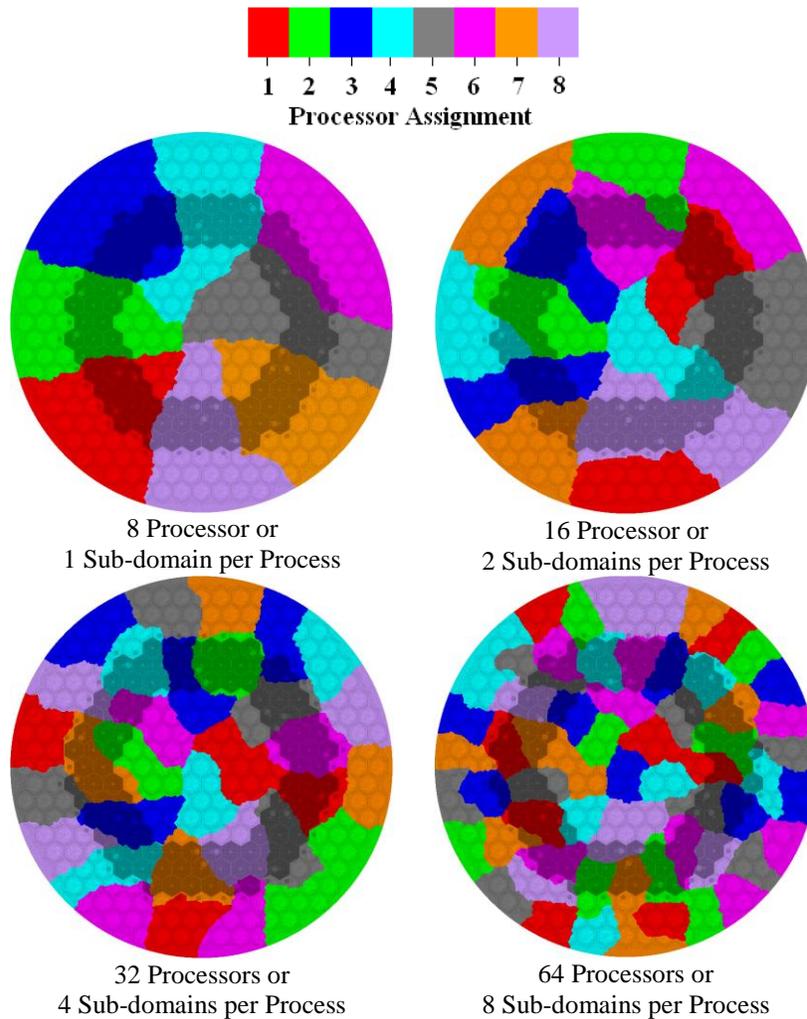


Figure 3.11 Example Multi-Domain Decomposition Approach to the VHTR

Even with this, however, a severe load imbalance can be observed because of the location of the assigned mesh. From Figure 3.11, the 8 processor decomposition is rather difficult to conceive having a bad load imbalance because each assigned domain effectively looks the same. In reality, if the direction of travel crosses from processor 8 to 4, these two processors can have substantially more trajectory data to process than processor 6 for the same direction. This situation only worsens as more sub-domains are defined such that at the 64 processor level, the same direction will cause all those sub-domains near the left and right hand side of the mesh to have very little work because of the small number of elements generated by the back projection. This is a consequence of the fact that a 0.01 cm^2 input criteria will be far greater than the actual area derived from intersecting all of the element surfaces.

To overcome this, we are studying whether assigning multiple sub-domains to a given process will yield a better load balance as depicted in Figure 3.11. Visually, using the 2 or 3 sub-domains per process level appears to show some level of improvement since it is difficult to find the clear example of work imbalance that would be expected at the 32 or 64 processor decomposition level (i.e. vastly different number of trajectories by direction by process).

Additional calculations are being prepared to assess the performance of this scheme along with the full back projection with the hope being improved scalability performance.

4 NODAL Solver Development

In response to the move to provide solution capabilities within UNIC that can solve problems on smaller sized parallel machines, minor amount of efforts were spent in FY2010 rebuilding the VARIANT capability [7] into UNIC. While the initial efforts are strictly focused on just the diffusion theory capability, it is not difficult to extend that work to a transport capability with time. The implementation is relatively easy if the old algorithm already present in DIF3D [8] is just reproduced. However, recent calculations using VARIANT on problems such as ZPR have indicated that the iterative schemes employed can break down quite severely which is observed as a rapid increase in computational effort. With the success of Krylov methods on SN2ND and MOCFE, along with their application in an even wider variety of engineering disciplines [17-18], the development effort on NODAL has focused a significant amount of effort on the impact of using Krylov (GMRES) on the response matrix equations

During the last fiscal year, we have examined the use of preconditioned Krylov methods as an alternative to the partitioned matrix accelerated red-black Gauss Seidel (RBGS) scheme that is presently used in VARIANT. The present work has shown that partitioned matrix acceleration is equivalent to preconditioning the RBGS solution and thus it is employed as the preconditioner to the GMRES algorithm. Since the partitioned matrix scheme can be considered a two-level p -multigrid preconditioner applied to a hierarchal set of trial functions, we refer to it as p -multigrid preconditioning.

4.1 Construction of the NODAL Preconditioner

To formulate the preconditioner, we start with the global set of response matrix equations having the form

$$[\mathbf{I} - \mathbf{R}]\mathbf{j} = \mathbf{q}, \quad (4.1)$$

where \mathbf{j} is the vector of partial currents at the nodal interfaces. A left preconditioner is defined by left multiplying the governing equation by the inverse of the preconditioner, thus if we take \mathbf{K} as the preconditioner, we have

$$\mathbf{K}^{-1}[\mathbf{I} - \mathbf{R}]\mathbf{j} = \mathbf{K}^{-1}\mathbf{q}. \quad (4.2)$$

As mentioned in the previous sections, an effective preconditioner approximates the inverse of the coefficient matrix, or $\mathbf{K} \approx \mathbf{I} - \mathbf{R}$, but requires many fewer floating-point operations to apply than the actual coefficient matrix.

Given the governing equation, we define a permutation matrix that reorders the unknowns in equations 4.1 and 4.2 such that the zero-order (i.e. spatially flat) term from each interface is gathered in \mathbf{j}_α while the higher order terms are contained in \mathbf{j}_β . Equation 4.1 can then be written as

$$\begin{bmatrix} \mathbf{I} - \mathbf{R}_{\alpha\alpha} & -\mathbf{R}_{\alpha\beta} \\ -\mathbf{R}_{\beta\alpha} & \mathbf{I} - \mathbf{R}_{\beta\beta} \end{bmatrix} \begin{bmatrix} \mathbf{j}_\alpha \\ \mathbf{j}_\beta \end{bmatrix} = \begin{bmatrix} \mathbf{q}_\alpha \\ \mathbf{q}_\beta \end{bmatrix}. \quad (4.3)$$

In response matrices derived from the variational nodal method, the unknowns at each nodal interface form an orthogonal series where only the zero-order term transports a net number of neutrons across the interface. Including the zero-order terms in the preconditioning matrix yields

$$\mathbf{K} = \begin{bmatrix} \mathbf{I} - \mathbf{R}_{\alpha\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \quad (4.4)$$

which has a strong physics based meaning. The inverse of equation 4.4 is found to be

$$\mathbf{K}^{-1} = \begin{bmatrix} [\mathbf{I} - \mathbf{R}_{\alpha\alpha}]^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}. \quad (4.5)$$

Left multiplying equation 4.3 by equation 4.5 yields

$$\begin{bmatrix} \mathbf{I} & -[\mathbf{I} - \mathbf{R}_{\alpha\alpha}]^{-1} \mathbf{R}_{\alpha\beta} \\ -\mathbf{R}_{\beta\alpha} & \mathbf{I} - \mathbf{R}_{\beta\beta} \end{bmatrix} \begin{bmatrix} \mathbf{j}_\alpha \\ \mathbf{j}_\beta \end{bmatrix} = \begin{bmatrix} [\mathbf{I} - \mathbf{R}_{\alpha\alpha}]^{-1} \mathbf{q}_\alpha \\ \mathbf{q}_\beta \end{bmatrix}. \quad (4.6)$$

As stated, employing equation 4.4 as a preconditioner in GMRES for equation 4.1 and using it as an acceleration scheme in VARIANT can be equivalence. To show this we assume red-black ordering (standard Cartesian solution scheme) and define $\mathbf{j}_\gamma^T = [\mathbf{j}_{\gamma r}^T, \mathbf{j}_{\gamma b}^T]$, $\mathbf{j}_\gamma^T = [\mathbf{q}_{\gamma r}^T, \mathbf{q}_{\gamma b}^T]$ with $\gamma = \alpha, \beta$ and thus

$$\mathbf{R}_{\gamma\delta} = \begin{bmatrix} \mathbf{0} & \mathbf{R}_{\gamma\delta}^{rb} \\ \mathbf{R}_{\gamma\delta}^{br} & \mathbf{0} \end{bmatrix} \quad \gamma, \delta = \alpha, \beta. \quad (4.7)$$

With this, we can rewrite equation 4.6 in the expanded form

$$\begin{bmatrix} \mathbf{j}_{\alpha r}^{i+1} \\ \mathbf{j}_{\alpha b}^{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{R}_{\alpha\alpha}^{rb} \\ -\mathbf{R}_{\alpha\alpha}^{br} & \mathbf{I} \end{bmatrix}^{-1} \left(\begin{bmatrix} \mathbf{0} & \mathbf{R}_{\alpha\beta}^{rb} \\ \mathbf{R}_{\alpha\beta}^{br} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{j}_{\beta r}^i \\ \mathbf{j}_{\beta b}^i \end{bmatrix} + \begin{bmatrix} \mathbf{q}_{\alpha r} \\ \mathbf{q}_{\alpha b} \end{bmatrix} \right) \quad (4.8)$$

$$\begin{bmatrix} \mathbf{j}_{\beta r}^{i+1} \\ \mathbf{j}_{\beta b}^{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{R}_{\beta\alpha}^{rb} \\ \mathbf{R}_{\beta\alpha}^{br} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{j}_{\alpha r}^{i+1} \\ \mathbf{j}_{\alpha b}^{i+1} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{R}_{\beta\beta}^{rb} \\ \mathbf{R}_{\beta\beta}^{br} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{j}_{\beta r}^i \\ \mathbf{j}_{\beta b}^i \end{bmatrix} + \begin{bmatrix} \mathbf{q}_{\beta r} \\ \mathbf{q}_{\beta b} \end{bmatrix}. \quad (4.9)$$

The superscript i is added to indicate the accelerated RBGS iterative procedure employed in VARIANT. Used in conjunction with either RBGS or GMRES, the inverse of $\mathbf{I} - \mathbf{R}_{\alpha\alpha}$ need not be evaluated exactly. Experience has shown that it is adequately approximated with a few iterations and increasing the number of iterations has only yielded marginal improvements in the convergence rate of RBGS or GMRES. This is of course expected and given that the minimal computational effort is very desirable, artificially increasing the number of iterations is generally unwise.

4.2 Comparison of GMRES to Existing VARIANT Algorithm

Figure 4.1 depicts the geometry used to compare the performance of the RBGS and GMRES solutions with and without p -multigrid preconditioning.

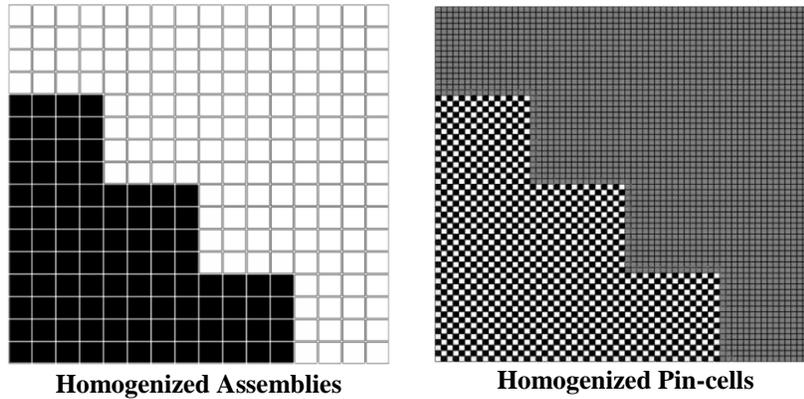


Figure 4.1 Quarter Core Configuration of Six LWR Fuel Assemblies and Reflector

In Figure 4.1, we show a geometrical configuration with homogenized fuel assemblies on the left and homogenized fuel pin cells on the right. A fixed-source, one-group cross section set was defined for each problem such that it emulates the fast group of a two-group LWR calculation. For the homogenized fuel assembly problem, Figure 4.2 shows the convergence of the L_2 norm of the residual vs. work units for RBGS (left) and GMRES (right).

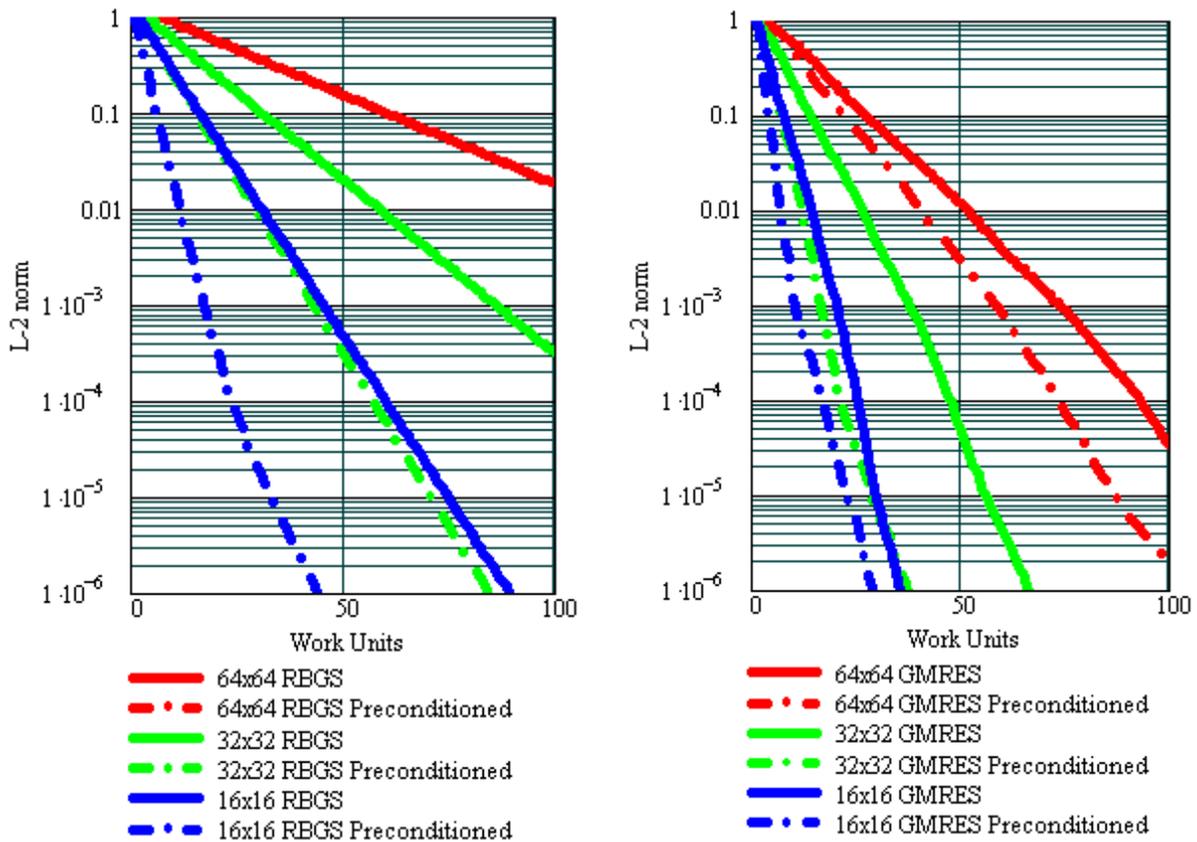


Figure 4.2 Convergence Rates of Red-Black Gauss-Seidel and GMRES for NODAL

One work unit is taken to be equal to the number of floating-point multiplications per unaccelerated iteration which is a reasonable measure of computational effort. Three

iterations were used on both RBGS and GMRES for inverting the preconditioner with every iteration of the two solution schemes. The successively coarser grid representations of the left hand geometry in Figure 4.1 were made assuming: 64x64, 32x32 and 16x16 where the response matrices use 1, 2, and 4 degrees of freedom (DOF) per nodal interface, respectively. The change in geometrical size tests the performance as the optical size of the node decreases and the change in surface degrees of freedom is done to maintain comparable levels of accuracy and overall work. From Figure 4.2, we can safely say that the GMRES performance is superior to RBGS at all three levels. More importantly, we can see that the p -multigrid is more effective as a preconditioner of GMRES than as an acceleration of RBGS. These results are consistent with several other problems we have executed and we anticipate all future work of NODAL will incorporate a GMRES solution capability.

4.3 Orthogonalized Matrix Aggregation

p -multigrid preconditioning is effective only for response matrices with orthogonal interface conditions which have more than one DOF per interface. In RBGS the acceleration cannot be applied when only one DOF is used per interface (i.e. fine mesh calculations) and in GMRES it makes the preconditioning much less effective (i.e. simple RBGS preconditioning). For fine mesh calculations, we find that the nodes become optically thin which causes RBGS solution methodologies to converge very slowly as evident from Figure 4.2. To circumvent this difficulty, we have developed a method of matrix aggregation and orthogonalization that converts $N \times N$ elemental matrices in to one response matrix with N orthogonal DOF per interface [19]. After applying this procedure, the forgoing p -multigrid method can be applied to the resulting response matrix equations.

To test the matrix aggregation and orthogonalization scheme in conjunction with p -multigrid preconditioning, we focus exclusively on the right hand geometry of Figure 4.1. The cross sections and dimensions are again representative of the fast group of a LWR calculation and we only consider convergence of the 64x64 grid. Applying response matrix aggregation and orthogonalization to obtain coarser grids of 32x32, 16x16, and 8x8, creates response matrices with 2, 4 and 8 DOF per nodal interface, respectively. The residual L_2 norm is plotted vs. work units in Figure 4.3. Note that the unpreconditioned GMRES results from Figure 4.3 are nearly identical to the unpreconditioned GMRES results of Figure 4.2 while the further p -preconditioned results are changed significantly. These results indicate that the p -multigrid preconditioning combined with the orthogonalized matrix aggregation can be effective; the 32x32 and 16x16 grid preconditioned solutions are significantly improved. However, at the coarsest grid (8x8), where the effect of matrix aggregation is the greatest, convergence without preconditioning is so rapid that applying the p -multigrid preconditioner has a negligible impact on the performance.

Using matrix aggregation requires back-substitution after convergence of each grid to update the partial currents internal to the aggregated domain. To account for this expense we calculate the “gain” in performance derived from using aggregation to a coarser grid in addition to using p -multigrid on that lower dimensional system by taking the following steps:

- 1) The work units required to obtain six orders of magnitude reduction in the L_2 norm are added to those required for the back-substitution.

- 2) The result from 1) is divided into the number of work units required for the same reduction in the L_2 norm for the unpreconditioned GMRES calculation on the fine grid.

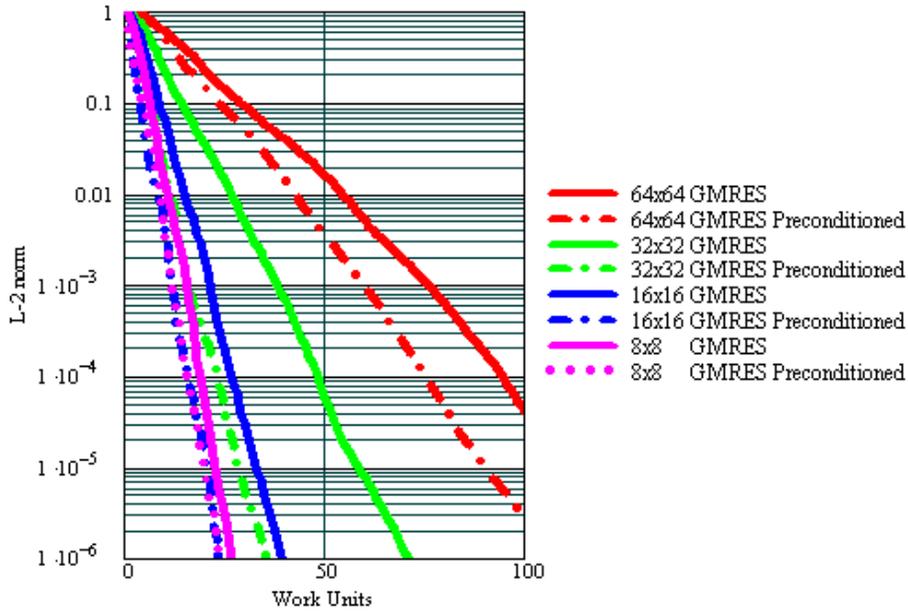


Figure 4.3 Impact of Combining p -multigrid and Orthogonalized Matrix Aggregation

This measure roughly shows the speed up achieved by preconditioning the fine mesh system with the combined scheme which is tabulated in Table 4.1. As can be seen, progressively larger gains occur as the mesh is coarsened, but as the aggregated domain becomes sufficiently large enough to negate the optical thinness of the resulting coarse meshes, the performance boost is constricted.

Table 4.1 Calculated Performance Gain Using Orthogonalized Matrix Aggregation

Aggregated Nodes	Gain without p -multigrid	Gain with p -multigrid
1	1.0	1.2
4	1.8	3.4
16	3.0	4.6
64	3.8	4.1

4.4 Summary Discussion

The implementation of NODAL into UNIC has been done at a much slower pace than either PN2ND, SN2ND, or MOCFE. In FY2009, the geometry models for all domains of interest were incorporated into UNIC and mapping capabilities to link the structured grid in NODAL with an unstructured grid in SN2ND/PN2ND/MOCFE were accomplished. Most of FY2010 focused on the research into the preceding preconditioners although we have completed the algorithmic implementation and are currently debugging the solver algorithm in UNIC. Based upon the previous analysis, we chose to implement the p -multigrid preconditioner for GMRES in PETSc in anticipation that NODAL will primarily be used as it

currently is in DIF3D. Only energy parallelism was incorporated into the NODAL solver development in anticipation that more energy groups would be routinely used by analysts which is a significant weak point of the DIF3D coding. With time, NODAL might be transitioned from a diffusion theory solver to a transport solver based upon the performance of SN2ND on small parallel machines and the focus of future work. At that time we will make a decision whether a discrete ordinates or spherical harmonics based transport methodology is developed at which time the preceding matrix aggregation preconditioning will become important.

5 Miscellaneous Components of the Neutronics Work

As was the case in previous years, a significant amount of time was spent developing components which are external to the UNIC solvers or algorithms usable by the solvers. Some of this work is a result of failures of the mesh generation tools while some is due to an inability of the Frameworks group of SHARP to provide a reasonably quick solution to difficult problems as was the case with the back projection algorithm in MOCFE. The remaining work involved creating necessary add-ons to UNIC to facilitate modeling and analysis of problems of interest to NEAMS while some was spent setting up and analyzing benchmark calculations. Overall, this work consumed almost a fourth of the total funding dedicated to neutronics which does not appear as a deliverable. The primary work non-solver related work for FY2010 can include IsoNXS, additional regression testing for the component libraries of UNIC, inclusion of Gauss-Lobatto-Tchebychev trial functions into element library, inclusion of an ability to evaluate the solution along a line traversing the domain, development of the BuildZPRmodel program, and creation of a new repository for containing MC²-3, UNIC, and the legacy ANL neutronics tools DIF3D and REBUS-3.

5.1 IsoNXS Storage of Tabulated Cross Section Data

The purpose of the IsoNXS work is to enable the coupled dynamics calculations in SHARP. The concept is derived from the legacy methodologies for coupling calculations where the cross section data is stored at a set of pre-evaluated criteria such as temperature, density, burnup, etc., which span the expected evaluation regime of the calculation. During the dynamics calculation, the cross section data is interpolated from the file to provide an accurate representation of the neutronics problem. The predecessor for IsoNXS was built for doing thermal-hydraulic feedback in REBUS-3, and was not appropriate for the parallel calculations to be carried out with UNIC. Given that the online cross section generation procedure will take several years to develop and implement in a parallel efficient manner, an algorithm similar to the legacy methodologies was felt necessary for use in UNIC in the short term. As a consequence, the storage procedure was redone in UNIC using the parallel HDF5 storage software. HDF5 is already part of UNIC where it is used to store the flux and power solution and is also part of the coupling methodology where it is used to read and store the mesh. While we do not envision doing parallel writing of cross section data with UNIC, the ability to do parallel reading is important and the primary purpose behind the new algorithm.

Tables 5.1 through 5.3 show the layout of the HDF5 storage format which is conceptually similar to the ISOTXS data structure, although noticeably different. In IsoNXS, we included additional variables to store the kinetics data for time dependent applications along with table evaluation information necessary to describe where each tabulated data point is stored.

Table 5.1. IsoNXS File Description: FileWide Group

<p>HDF5 “test1a.h5” Group “/” Group “/FileWide” EnergyBounds (NumGroups+1) GramAtomMasses(NumIsotopes) IsotopeNames(NumIsotopes) LibraryProperties(10): NumGroups, NumIsotopes, MaxScatteringOrder, MaxScatteringBlockLW, NumTableProperties, StandardizedScattering, N2nProductionBased, MaxNumFamilies, IsoNXS_Version, IsoNXS_yyyymmdd PropertyNames(NumTableProperties): (e.g. PackingFraction, Burnup, Temperature) ReactionRateNames(IsoNXS_NUM_KNOWN_REACTION_RATES) ScatterReactNames(0:3) = (“Scat_Total”, “Scat_Elastic”, “Scat_Inelastic”, “Scat_N2n”) Velocities(NumGroups)</p>

Table 5.2. IsoNXS File Description: Isotope Header Group (repeated for each Isotope)

<p>Group “/u235b” Group “/Header” DecayConstants(NumFamilies) DelayChi(DelayFissionCutoff, DelayChiWidth, NumFamilies) HeaderInt(7): NumTablePoints, MaxScatteringBlock, ChiWidth, FissionCutoff, NumFamilies, DelayChiWidth, DelayFissionCutoff HeaderReal(2): EnergyCapture, EnergyFission NumAuxiliary(IsoNXS_NUM_KNOWN_REACTION_RATES) NumScatterAux(0:3)=MaxScatteringOrder OffsetToBand(NumGroups+1, MaxScatteringOrder, 0:3) PropertyValue(NumTableProperties, NumTablePoints) StartingGroup(NumGroups, MaxScatteringOrder, 0:3) TotalChi (FissionCutoff, ChiWidth)</p>

Table 5.3. IsoNXS File Description: Isotope Reaction Groups (repeated for each Isotope)

<p>Group “/u235b” Group “/Reactions” Alpha_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_ALPHA), NumTablePoints) DelayNu_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_DELAYNU), NumTablePoints) Deuterium_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_DEUTERIUM), NumTablePoints) DirectDiff_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_DIRECTDIFF), NumTablePoints) Fission_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_FISSION), NumTablePoints) Ngamma_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_NGAMMA), NumTablePoints) Nu_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_Nu), NumTablePoints) Proton_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_PROTON), NumTablePoints) Scat_Elastic_XSDDataBlock(MaxScatteringBlock(1), NumTablePoints) Scat_Inelastic_XSDDataBlock(MaxScatteringBlock(2), NumTablePoints) Scat_N2n_XSDDataBlock(MaxScatteringBlock(3), NumTablePoints) Scat_Total_XSDDataBlock(MaxScatteringBlk(0), NumTablePoints) Total_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_TOTAL), NumTablePoints) Transport_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_TRANSPORT), NumTablePoints) Tritium_XSDData(NumGroups, NumAuxiliary(IsoNXS_IDNum_TRITIUM), NumTablePoints) Group “/ReactionsIndex” Alpha_TablePointIndex(NumTablePoints) DelayNu_TablePointIndex(NumTablePoints) Deuterium_TablePointIndex(NumTablePoints) DirectDiff_TablePointIndex(NumTablePoints) Fission_TablePointIndex(NumTablePoints) Ngamma_TablePointIndex(NumTablePoints) Nu_TablePointIndex(NumTablePoints) Proton_TablePointIndex(NumTablePoints) Scat_Elastic_TablePointIndex(NumTablePoints) Scat_Inelastic_TablePointIndex(NumTablePoints) Scat_N2n_TablePointIndex(NumTablePoints) Scat_Total_TablePointIndex(NumTablePoints) Total_TablePointIndex(NumTablePoints) Transport_TablePointIndex(NumTablePoints) Tritium_TablePointIndex(NumTablePoints) MaxScatteringBlock(Type)= OffsetToBand(NumGroups+1, MaxScatteringOrder, Type) –OffsetToBand(1, 1, Type)</p>

As can be seen, the data is stored in an isotopic basis which is consistent with the way the data is retrieved during the material homogenization process. All functions have been written to minimize the involvement of user coding with the underlying structures that access the data on the file (file groups, hyperslabs, etc.). Regression tests have been implemented for most parts of the IsoNXS library.

In addition to creating the IsoNXS file format, we also had to update the ISOTXS reader built into the UNIC code. The old version of the ISOTXS file reader/writer was not capable of importing data files larger than 70 groups on BlueGene/P due to memory constraints and poor choices made in creating the reader. To facilitate the ZPR calculations carried out in this report, the ISOTXS file reader/writer was updated to be more memory efficient. While the ISOTXS reader is already included in UNIC, the IsoNXS reader has not been fully implemented since it was awaiting the completion of the revised SN2ND solver which is the primary development purpose for IsoNXS.

5.2 Gauss-Lobatto-Tchebychev Finite Element Capability

The element library of UNIC was extended with the Gauss-Lobatto-Tchebyshev (GLT) finite elements. Trial functions for the bar, quadrilateral, triangle, brick, tetrahedron, and prism were added and verification tests were created. For higher than cubic order finite elements, these functions produce coefficient matrices in SN2ND which have lower condition numbers. A brief study of the one-dimensional set of functions is sufficient to understand why.

To begin, the coordinate vertices of a Lagrangian finite element are specified to be equally spaced on the one-dimensional space while the coordinate vertices of the GLT elements are given by the roots of the n -th order Tchebychev polynomial

$$T_n(x) = \cos[n \cos^{-1}(x)]. \quad (5.1)$$

Figure 5.1 compares the GLT and Lagrangian shape functions for 5th and 10th order trial functions which are noticeably different. As the trial function order increases, the GLT trial functions are observed to obtain their maximum at the coordinate vertex while the Lagrangian functions progressively display maximum values near the end points of the domain. While the fundamental span of the basis functions is the same, the linear dependence between the functions (they are not orthogonal) is such that GLT shows substantial improvements in the spectral radius of the coefficient matrix at fourth order (tests performed in MathCAD).

For bar, quadrilateral, and brick elements, these trial functions can be formed using a product of the one-dimensional functions. For triangle, prism, and tetrahedral functions, we used a more complicated procedure to generate them and thus only implemented sixth order trial functions in UNIC. Since CUBIT does not generate meshes with these trial functions and MOAB doesn't support them we have yet to test them out. We have also not implemented them into the NODAL code which is likely the best way to quickly utilize them since it utilizes the data structures contained within UNIC and can build the appropriate meshes. Ideally these trial functions will allow better modeling of the homogenized assembly level calculations which SN2ND has not performed well on up to this point. While higher order Lagrange trial functions were attempted, the degradation in performance was so severe that any improved accuracy could not be justified.

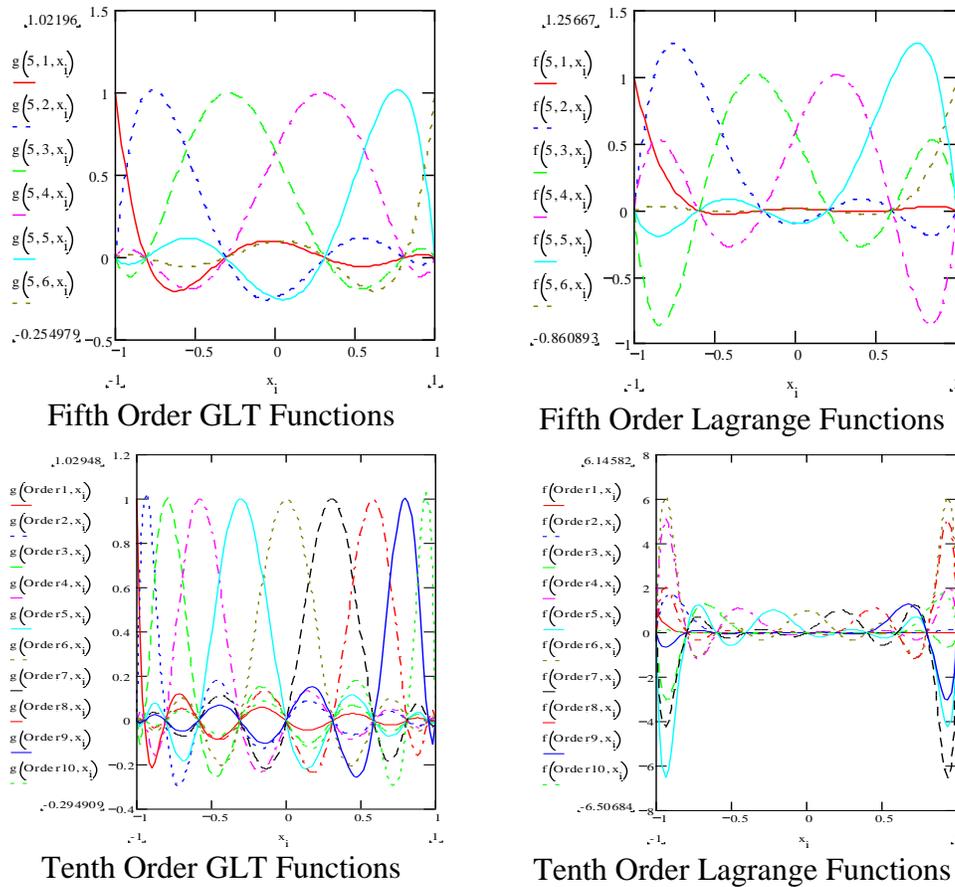


Figure 5.1 One-Dimensional GLT and Lagrangian Finite Element Basis Functions

5.3 Evaluating the Solution along a Traversing Line

The long awaited ability to evaluate the flux solution along a line traversing the domain was added to UNIC. The primary reason for the delay was the difficulty involved in implementing an efficient algorithm into UNIC. Also, given that SN2ND has relied primarily upon quadratic elements, the existing line evaluation capability within VISIT is sufficient to provide an accurate solution along a traversing line. While initially thought to be a frameworks related issue, the storage of the flux solution can easily be an overwhelming amount of data to duplicate (in frameworks) and then process for a significant number of groups. Given that the detailed flux solution is rarely important – normally we are only interested in power and capture rates – we desire a capability to evaluate the flux solution along the traverses during execution to minimize the outputted storage.

Given that we can identify which elements in the domain are intersected by the traversing line, we must devise an efficient method to evaluate the flux solution. Since the interpolation functions for every finite element are in the reference system (i.e. r-s-t, not x-y-z), we must use a root finding technique to identify the location of the intersected coordinate along the line (x-y-z) in the reference system of each finite element (r-s-t). From there we can evaluate the interpolation functions. The most accurate way to do this is to use a root finding technique, such as Newton, which requires multiple evaluations of the shape functions and can become computationally expensive. We also studied the capability used by VISIT since previously

experience using it provided a good solution along the line capability. Figure 5.2 graphically depicts the sub-element methodology incorporated by VISIT [29].

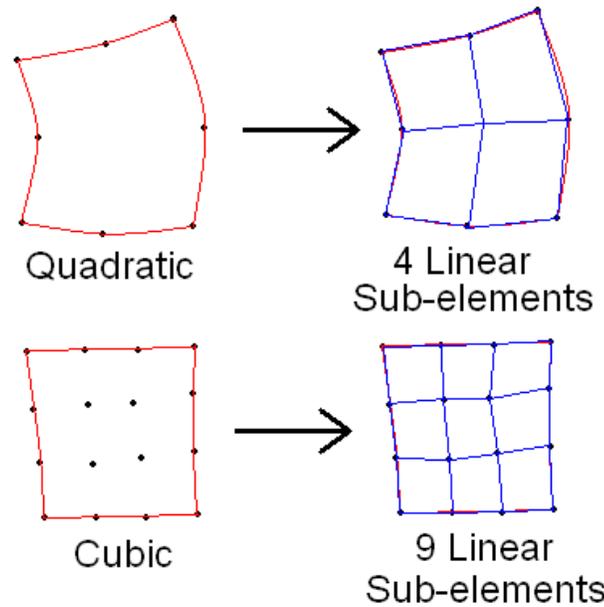


Figure 5.2 VISIT Sub-element Approximation of a Finite Element

As can be seen, VISIT uses a series of linear interpolating finite elements to interpolate between the solution points. Starting with the two finite elements shown in Figure 5.2, we show a typical crossing line as a series of x's in Figure 5.3 that would result from a line traversing the domain. Note that both elements are geometrically the same and that a linear mapping is utilized to minimize the error associated with the mapping itself. This limits the error produced by the subelement approximation, since it would use an easily solved linear interpolation algorithm to identify the coordinate point in the reference system. Rather than just testing the x points, we sampled the entire finite element with a large set of equally spaced points.

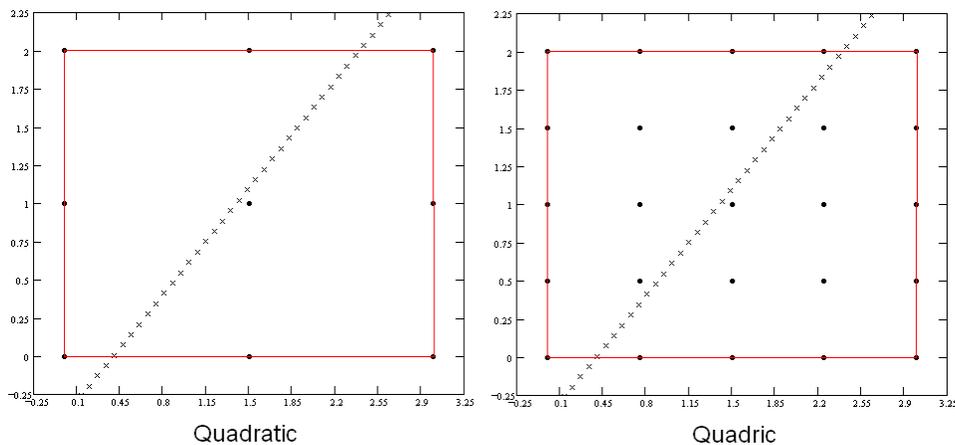


Figure 5.3 Quadratic and Quadric Finite Elements and a Flux Traverse (x's)

A plausible flux solution was assumed in both finite elements. Figure 5.4 shows the flux solution evaluated at the series of test points which are generally smooth and consistent with the order of the finite element in question.

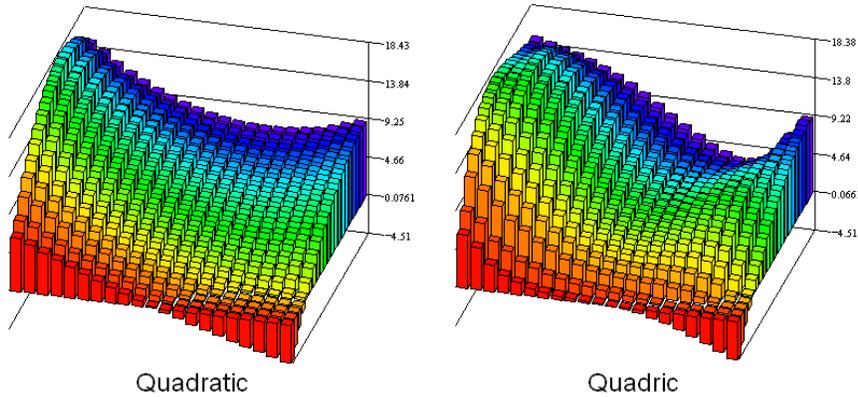


Figure 5.4 Assumed Quadratic and Quadric Flux Solutions

Reproducing the sub-element technique in MathCAD, we obtained the solution data and calculated its error with respect to the assumed flux solution as shown in Figure 5.5.

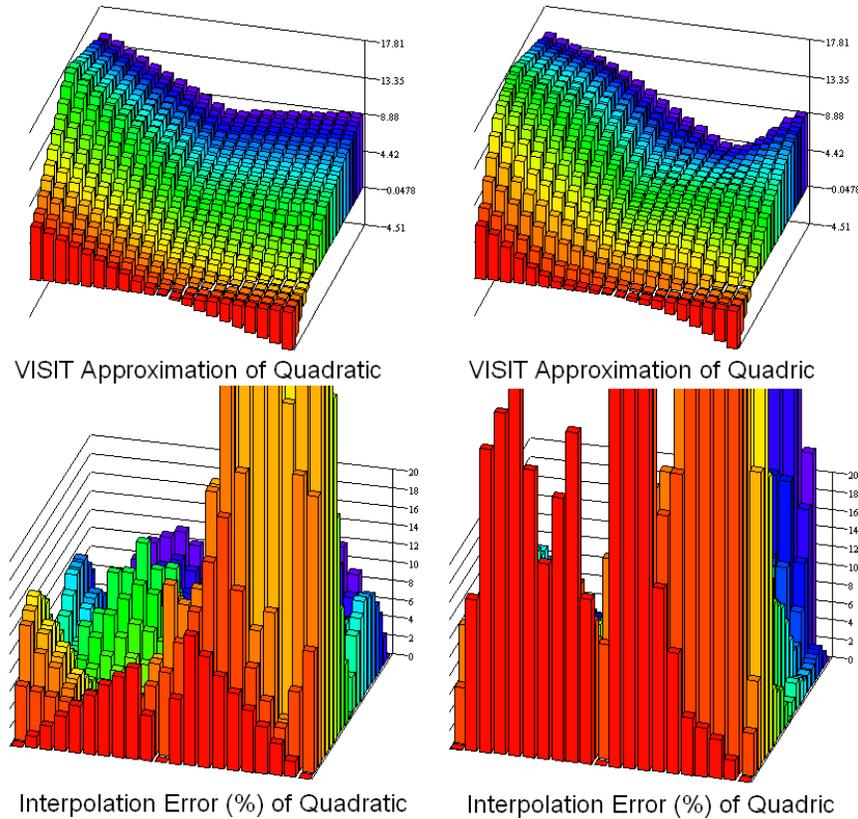


Figure 5.5 Solutions and Calculated Error of the Sub-element Scheme

We note that there is an observable reduction in the smoothness of the flux, but the approximation does not qualitatively look bad. Of course, inspection of the error generated by

the sub-element representation demonstrates that there is a considerable amount error, in some cases above 100%. While a majority of the domain is less than 20% error in both cases, the irregularity of the error is cause for great concern with regard to a flux traverse. It is also important to note that as the interpolation of the element is increased beyond linear, the sub-element derived error increases significantly (not shown for brevity). Given our desire to provide the most accurate solution possible, we utilized the Newton root finding technique to identify the location of the evaluation point within each element.

To determine which elements are crossed, we make use of many parts of the MOCFE solver since each traversing line is similar to a MOC trajectory. The difference of course is that in MOCFE we used a complicated procedure to define the starting location of the trajectory on the domain. Given that point, the entire MOCFE coding is built around the concept of following the trajectory as it progresses through the domain from the incident intersection point through all elements that are crossed along the trajectory to an exiting point on the domain. For the evaluation technique, given there are relatively few trajectories, it was simpler from a programming perspective to just check all surfaces of all elements to determine the list of intersected elements. The same triangle intersection algorithm in the 3-D MOCFE ray tracer and the line intersection algorithm in the 2-D MOCFE ray tracer were used for this work.

Also, rather than specify a fixed number of points along the line, a variable number of points within each element is generated depending upon the user-defined point density desired along the line between the intersecting points on each element. The Newton algorithm is used to identify the position of that point within the reference element and thus the point to pick for evaluating the trial functions of each element. These points are collected on a mesh block basis such that only a single call per block to ElementLibrary is needed to evaluate the shape functions. The computational effort of determining the flux solution is then reduced to a simple vector dot product for each evaluation point within the block. The data is reduced to the root process for export to an ASCII file format importable directly into EXCEL or gnuplot for easy visualization.

This capability was the primary means by which the ZPR foil activation results were obtained as described later in this report. In addition to that purpose, it can also be used to study ray effects and mesh convergence. Figure 5.6 shows the highest energy group flux mesh convergence behavior of the SN2ND solver for a traverse running just inside of the outer surface of the ZPR matrix half where the matrix halves meet while Figure 5.7 shows the mesh convergence for a traverse running through the two drawers at the center of the assembly. Insets are included to show the slight variation in the solution between a 299,276 vertex mesh and a 408,112 vertex mesh where the 103,684 vertex mesh is a linear mesh.

The primary difference between the two most refined meshes in these figures is the number of elements used in the axial direction. In problems that display more error with respect to mesh refinement, we have observed significant differences in the solution. A more reliable means to study the impact of mesh refinement with SN2ND is to look at the element-wise odd-parity P_1 fluxes. Because SN2ND uses a continuous even-parity flux approximation, the discontinuities appear in the odd-parity flux. Experience has shown large discontinuities in the odd-parity flux indicate locations where mesh refinement is necessary. With time we intend to add the odd-parity flux components to the evaluation output files such that they can be viewed like the scalar flux plots above.

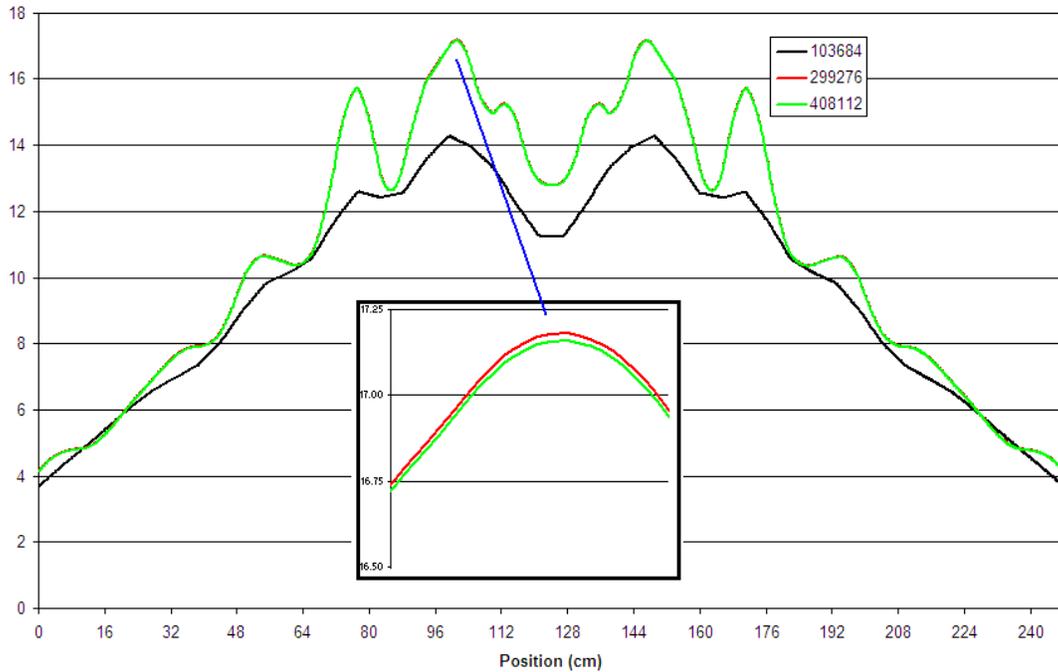


Figure 5.6 ZPR-6/7 Matrix Edge Group 1 Flux Plot for Loading 106 (Mesh Convergence)

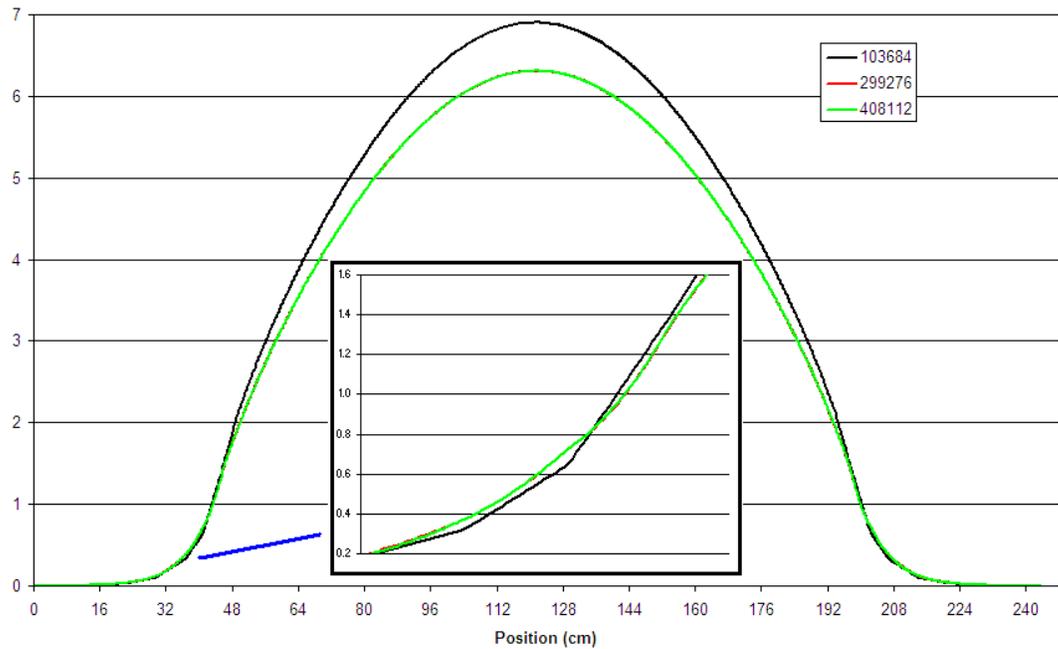


Figure 5.7 ZPR-6/7 Center Drawer Group 1 Flux Plot for Loading 106 (Mesh Convergence)

Figure 5.8 shows the often named ray effect phenomena accompanying all discrete ordinate methods for the 408,112 vertex mesh. In this case we purposely choose the traverse used in Figure 5.6 since we know it will yield substantial ray effects. While the eigenvalue is nearly identical between these two calculations, the flux solution exhibits wild, non-physical oscillations which change dramatically as the angular cubature is refined. The only reliable

way to remove the ray effects in Figure 5.8 is to increase the angular cubature which would likely require well past $P_{11}T_{11}$ to obtain a reasonably smooth flux shape.

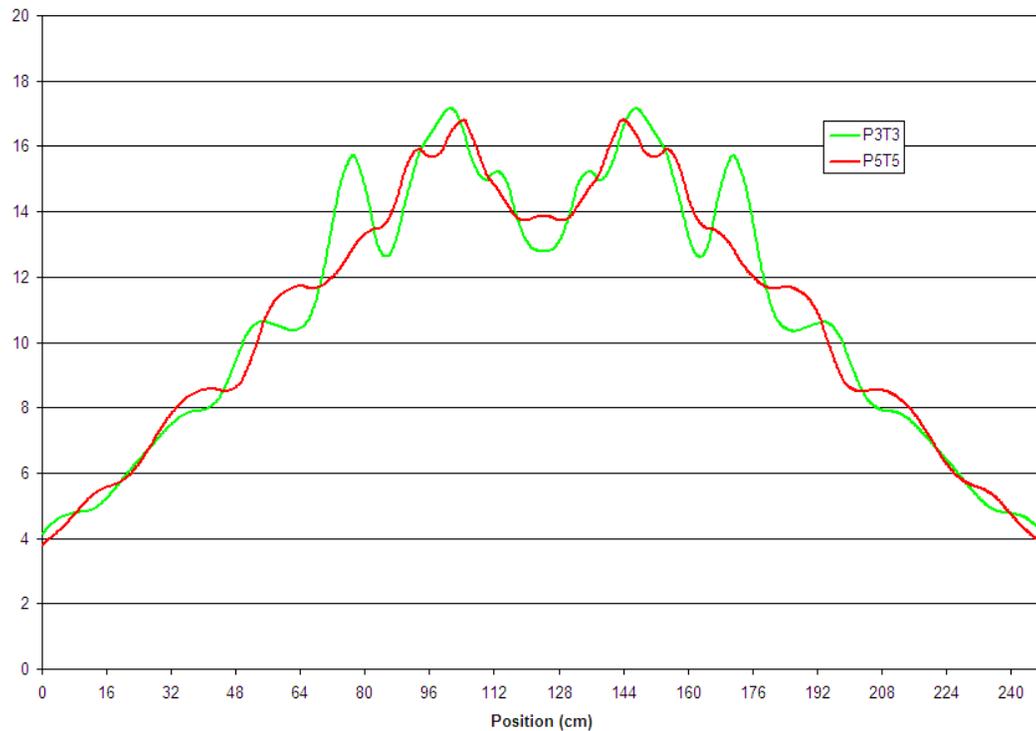


Figure 5.8 ZPR-6/7 Matrix Edge Group 1 Flux Plot for Loading 106 (Ray Effects)

5.4 BuildZPRmodel and MergeMesh, Programs to Generate Input Geometry

As part of the verification and validation requirements of UNIC, we have chosen to study the ZPR series of experiments. Unlike many of the numerical benchmarks on which we have used UNIC, the ZPR experiments provide a very clean geometry (well known material definitions) and include numerous reactor physics measurements that we can use to validate the steady state solver solutions, such as activation foil measurements, and various measurements that we can use to validate the kinetics solver capability.

As one would expect, trying to build a geometry model by hand that consists of numerous non-repeating structures such as that seen in Figure 5.9 is very prone to human errors. To combat this issue, the BLDVIM tool [20] was created previous to the SHARP project. Attached to BLDVIM is a library of materials defining the compositions (batch average) and dimensions of each known plate and drawer that was loaded into the ZPR experiments. Given simple user ASCII input to define the drawer-wise geometry loading of a ZPR experiment (ZPR-3, ZPR-6, ZPR-9, and ZPPR), BLDVIM will build the plate-by-plate geometry for use in the VIM continuous energy Monte Carlo code.

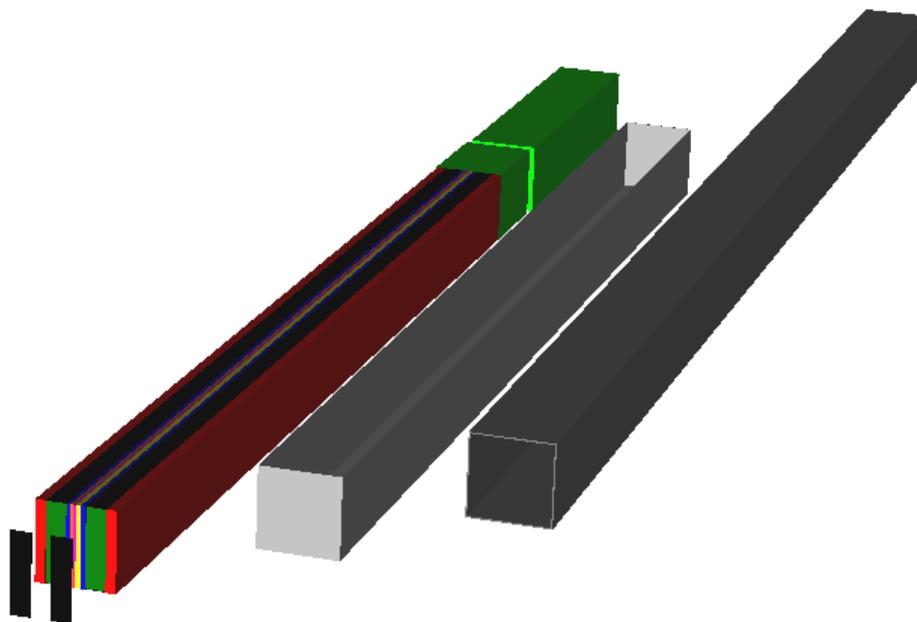


Figure 5.9. Single ZPR-6 Core Drawer

Monte Carlo modeling of a ZPR can provide very accurate solutions because there is virtually no thermal gradient and there are relatively few compositions. With a deterministic methodology such as SN2ND, the problem setup is much more complicated because the cross section data has to be generated on a representative cell. If we consider that unique cross section data is needed for every plate in every unique drawer, then we can easily end up with an order of magnitude more compositions than that used in the Monte Carlo model. As a consequence, the deterministic approach requires geometry and mesh generation techniques that allow easy mapping of these materials to the geometric zones and minimize potential user errors. The existing mesh generation and frameworks tools are quite clumsy when it comes to mapping the materials (a by-hand procedure) and thus using CUBIT on ZPR is quite an endeavor. Without an available capability, the BuildZPRmodel code was created to fulfill the need.

The original version of BuildZPRmodel (ZPRtoNODAL) was created in FY2009. In FY2010, we modified it so that it would create slab geometry input for generating cross sections with MC²-3. In that process we had BuildZPRmodel define material compositions with the appropriate mapping to the MC²-3 generated isotopes and material mapping to the user-defined ZPR model. The final change was to add an unstructured mesh generation algorithm usable in UNIC (previous version only built a structured NODAL geometry). Because of the complexity of the tool and the likelihood that an equivalent capability will not be available soon, we have also invested a significant amount of time setting up nightly verification tests for BuildZPRmodel.

With respect to deterministic modeling, the primary purpose of BuildZPRmodel is to allow the reactor analyst to define the homogenization scheme and simultaneously build the appropriate MC²-3 input decks. This homogenization scheme requires that the user define a homogenization model on a drawer by drawer basis where the isotopes from one drawer can

be linked to another drawer in the model. Figure 5.10 shows an example of the homogenization approaches we have applied to date in BuildZPRmodel.

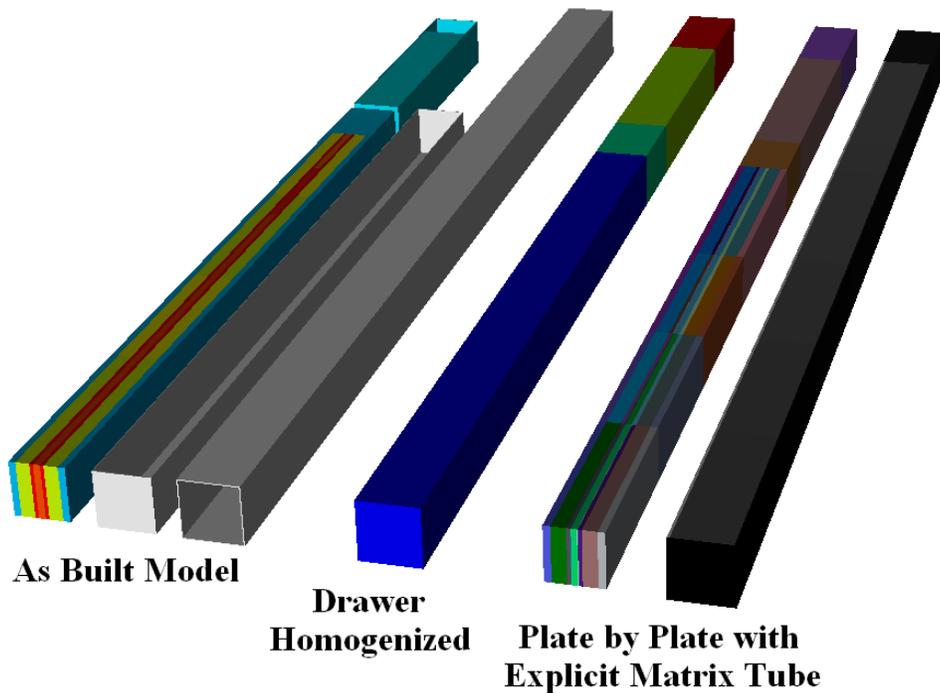


Figure 5.10. Example Homogenization Models

In Figure 5.10, the “as built model” shows the geometry where the plates are separated from the front drawer (center) and both are separated from the matrix tube (right). In the “drawer homogenized” model, the plate structure is completely homogenized maintaining only the primary axial heterogeneity distinguished by the active core plates, axial depleted uranium blankets, and the empty matrix tube. In the plate by plate homogeneous model, each plate is preserved while the drawer and matrix tube are smeared together to form a surrounding box.

The drawer homogenized model is consistent with the cross section generation schemes applied in reactor physics for the last 30 years. As a consequence, we have typically observed very good results when comparing the eigenvalues and reaction rates with Monte Carlo and the experimentally measured values so long as the problem is well within a valid regime of the typical cross section generation procedure. The weakness of course is that the global flux solution derived from the drawer homogenized approach does not exactly reproduce the plate-by-plate flux solution (flux reconstruction does not help because the slab or 2-D slice is not accurate) and thus resolving the foil measurements is somewhat erroneous.

Because SHARP has been focused on a neutronics capability for heterogeneous modeling, i.e. plate by plate, we have been studying how to accomplish this on the ZPR problems since we have such good experimental data. However, without the new multigrid preconditioner, SN2ND cannot accurately treat the ZPR with enough space-angle-energy resolution and the solutions are generally less accurate than the drawer homogenized solutions. The additional fact that we have to learn how to properly generate cross sections only complicates the matter

further. With a tool like BuildZPRmodel, we can at least test the various homogenization methodologies out on smaller problems (i.e. single drawers) without requiring an inordinate amount of effort being expended in setting up the input information.

As mentioned, the ZPR geometry is fundamentally not a structured geometry. While the drawers are arranged in a lattice, slight variations in the plate thicknesses combined with the inclusion of control rod positions, fission chambers, thermal couples, and movable drawer positions (another type of control rod), prevents a structured geometry tool such as PARTISN from being able to model a ZPR problem without generating hundreds of millions, if not billions of mesh cells. Figure 5.11 shows an example of the unstructured mesh capability we added to BuildZPRmodel which uses 1.2 million unstructured brick elements to describe the semi-homogenized geometry.

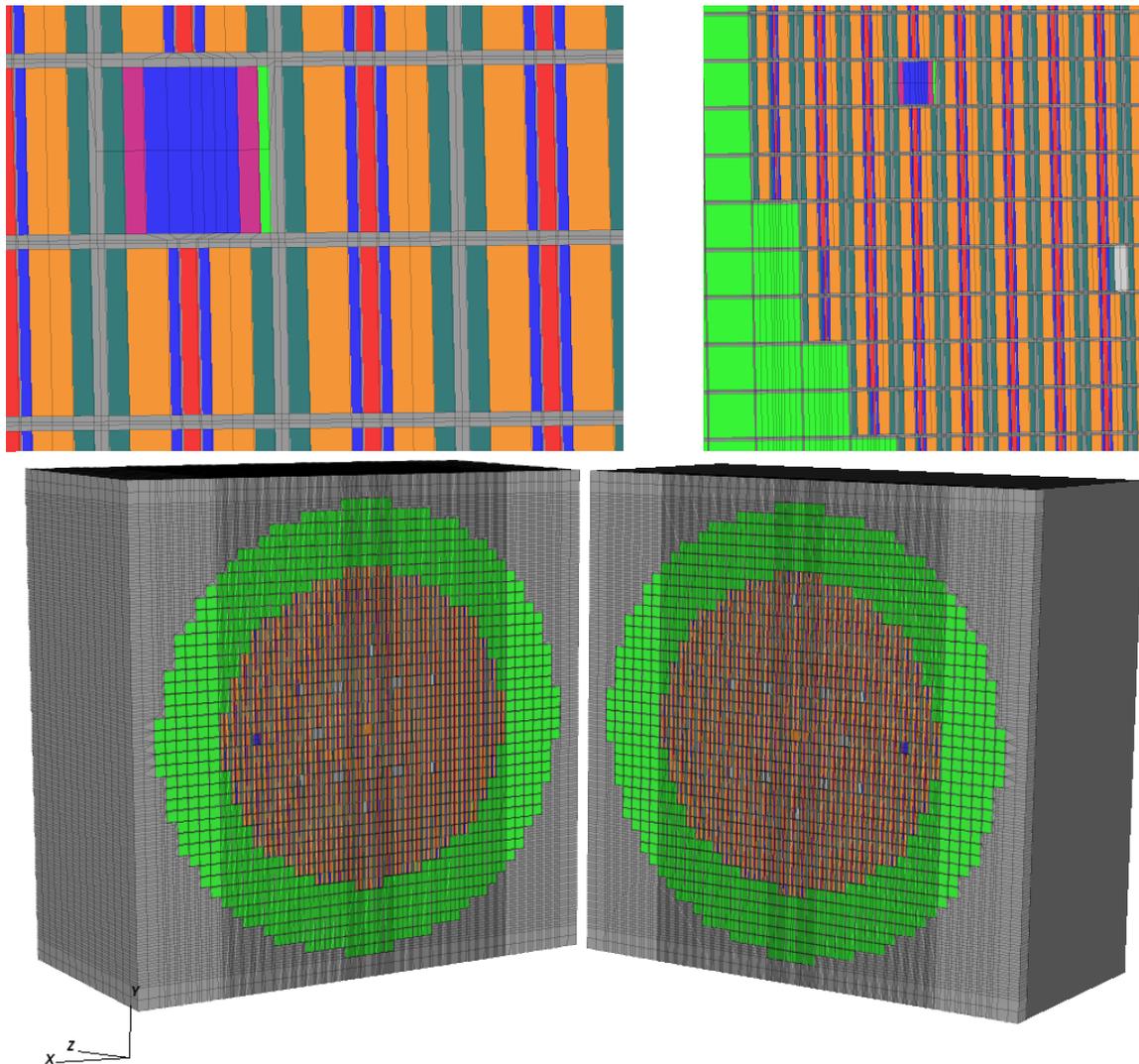


Figure 5.11. Full Core ZPR-6/7 Model Built with BuildZPRmodel

In order to create this mesh, we had to add a new mesh merging algorithm to UNIC. Some time had already been spent in SHARP making a similar tool, but it was not capable of handling a Cartesian geometry or tracking the boundary conditions and material assignments.

The algorithm in UNIC is not terribly dissimilar from the concept of that tool, but it does not have to assume a regular grid and allows the user to easily define the placement of the submeshes. Many of reactor problems are now created using the UNIC mesh merging function because CUBIT is incapable of generating the full geometry mesh such as that seen in Figure 5.12. This mesh of the full core VHTR geometry was used for the MOCFE scaling studies discussed in the previous section.

Combining this feature with the extrusion tool developed for UNIC in FY2007 allows us to build explicit full core geometry reactor problems which we were not capable of doing in previous years with existing SHARP tools. Figure 5.13 shows some example input for the merge mesh routine. Note that in BuildZPRmodel, the merge mesh routine is a simple function call since it was built using existing structures of UNIC. The input in Figure 5.13 is relatively straightforward indicating the geometry type (hexagonal) and the pitch. The second line is the most important as it specifies the number of user created mesh files and the size of the grid to form in the X and Y directions. The grid of XY positions follows the list of mesh file names where the numbers give the index number of each user mesh file. Note that this approach imposes the entire X-Y grid be given and that 0s be used to eliminate those positions where mesh placement is not desired. All of the user meshes except for the barrel (mesh files 1 and 2) are assumed to be centered about the origin. The barrel should be offset such that its selected position yields the desired result.

Combined, these new tools required several months of development which could have alternatively been used on solver development. Regardless, we can now properly generate geometry and compositions for the ZPR calculations in addition to being able to generate geometries for heterogeneous geometries in both two- and three-dimensions.

5.5 New Repository and Component Verification of UNIC

The ongoing process of adding code verification also consumed some effort this year. In the FY2009 we setup the BuildBot tool [30] to carry out nightly regression tests for some basic parts and top-down tests for the PN2ND and SN2ND solvers. In FY2010, we extended the regression tests to include GaussLibrary, ElementLibrary, MeshSphere, NonZeroLibrary, along with BuildZPRmodel components and IsoNXS. We also added top down tests for the two-dimensional MOCFE solver and have started to develop tests for the three-dimensional MOCFE solver. These tests pointed out several minor mistakes throughout the various subroutines and functions which were updated as part of this work thereby making UNIC a more reliable tool.

To carry out the development of the inline cross section generation, we must be able to merge and distribute the MC²-3 and UNIC tools. The rules for distributing MC²-3 have been well established such that it cannot be considered an open source or widely distributable software. With respect to the SHARP repository currently hosted by MCS to contain the UNIC and NEK codes, including a copy of MC²-3 would violate the outstanding prescribed export control rules assigned to MC²-3. To allow the inline cross section generation code to be developed, it was felt best to separate the developmental version of UNIC from the SHARP repository and create a new one hosted in a more secure and controlled location where MC²-3 can actually be stored. Because of the change in focus, we have also taken the initiative to include the legacy neutronics tools DIF3D and REBUS-3 into this repository to allow for a single development point of all SHARP related activity.

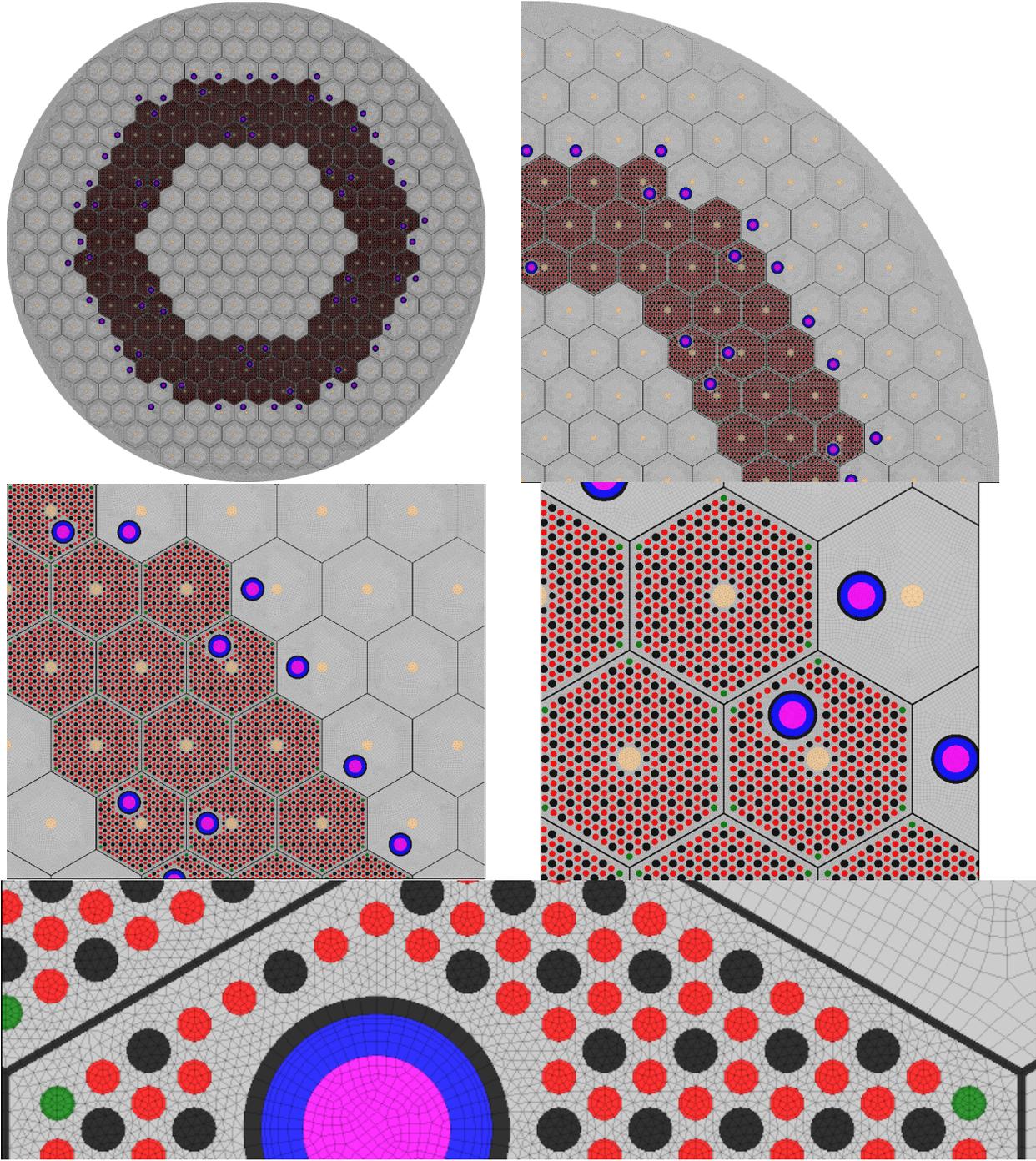


Figure 5.12. Example VHTR Mesh Built Using the New Merge Mesh Routine

<pre> Hexagonal 36.296900000 0 26 11 6 VHTR_Barrel_30.ascii VHTR_Barrel.ascii VHTR_CentralR.ascii VHTR_CentralR_c000.ascii VHTR_CentralR_c030.ascii VHTR_CentralR_s030.ascii VHTR_Control_000.ascii VHTR_Control_060.ascii VHTR_Control_120.ascii VHTR_Control_180.ascii VHTR_Control_180_c030.ascii VHTR_Control_240.ascii VHTR_Control_300.ascii VHTR_OuterCTRL_000.ascii VHTR_OuterCTRL_060.ascii VHTR_OuterCTRL_120.ascii VHTR_OuterCTRL_180.ascii VHTR_OuterCTRL_180_c000.ascii VHTR_OuterCTRL_240.ascii VHTR_OuterCTRL_240_c030.ascii VHTR_OuterCTRL_300.ascii VHTR_OuterR.ascii VHTR_OuterR_c000.ascii VHTR_OuterR_c030.ascii VHTR_Standard.ascii VHTR_Standard_c000.ascii 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 20 22 22 0 0 0 0 0 0 0 11 25 19 22 22 0 0 0 0 0 5 9 25 25 22 22 22 0 0 0 5 3 3 25 25 12 17 22 0 0 6 4 4 4 4 26 26 18 23 23 1 </pre>	<pre> Hexagonal 36.296900000 0 26 21 21 VHTR_Barrel_30.ascii VHTR_Barrel.ascii VHTR_CentralR.ascii VHTR_CentralR_c000.ascii VHTR_CentralR_c030.ascii VHTR_CentralR_s030.ascii VHTR_Control_000.ascii VHTR_Control_060.ascii VHTR_Control_120.ascii VHTR_Control_180.ascii VHTR_Control_180_c030.ascii VHTR_Control_240.ascii VHTR_Control_300.ascii VHTR_OuterCTRL_000.ascii VHTR_OuterCTRL_060.ascii VHTR_OuterCTRL_120.ascii VHTR_OuterCTRL_180.ascii VHTR_OuterCTRL_180_c000.ascii VHTR_OuterCTRL_240.ascii VHTR_OuterCTRL_240_c030.ascii VHTR_OuterCTRL_300.ascii VHTR_OuterR.ascii VHTR_OuterR_c000.ascii VHTR_OuterR_c030.ascii VHTR_Standard.ascii VHTR_Standard_c000.ascii 0 0 22 22 22 22 22 22 22 0 0 0 0 0 0 0 0 0 0 0 0 0 22 22 22 22 22 22 22 22 22 0 0 0 0 0 0 0 0 0 0 0 22 22 22 22 19 19 21 21 22 19 22 22 22 0 0 0 0 0 0 0 0 22 22 21 21 25 10 25 25 25 13 19 22 22 22 0 0 0 0 0 0 0 22 22 22 7 25 25 25 12 25 25 25 25 17 22 22 0 0 0 0 0 0 22 22 14 25 25 25 25 10 10 25 25 25 9 17 22 22 0 0 0 0 0 22 22 14 25 25 25 3 3 3 3 3 25 25 25 19 22 22 0 0 0 0 22 22 21 25 13 12 3 3 3 3 3 3 9 10 25 19 22 22 0 0 0 22 22 21 12 25 12 3 3 3 3 3 3 3 3 9 25 25 22 22 0 0 0 22 22 25 25 25 3 3 3 3 3 3 3 3 25 25 12 17 22 0 0 0 22 22 14 25 25 3 3 3 3 3 3 3 3 3 25 25 17 22 22 2 0 0 22 14 8 25 25 3 3 3 3 3 3 3 3 3 25 25 22 22 0 0 0 22 22 25 25 13 3 3 3 3 3 3 3 3 8 25 8 16 22 22 0 0 0 22 22 15 25 7 13 3 3 3 3 3 3 8 9 25 16 22 22 0 0 0 0 22 22 15 25 25 25 3 3 3 3 3 25 25 25 17 22 22 0 0 0 0 0 22 22 14 13 25 25 25 7 7 25 25 25 17 22 22 0 0 0 0 0 0 22 22 14 25 25 25 25 8 25 25 25 10 22 22 22 0 0 0 0 0 0 0 22 22 22 15 9 25 25 25 7 25 16 16 22 22 0 0 0 0 0 0 0 0 22 22 15 22 22 16 16 15 15 22 22 22 22 0 0 0 0 0 0 0 0 0 22 22 22 22 22 22 22 22 22 22 0 0 0 0 0 0 0 0 0 0 0 22 22 22 22 22 22 22 22 22 0 0 0 0 0 0 0 0 0 0 0 0 22 22 22 22 22 22 22 22 0 0 </pre>
<p>Hexagonal VHTR with 30 degree symmetry</p>	<p>Full core hexagonal VHTR</p>

Figure 5.13. Example Input Specification for Merge Mesh Routine in UNIC

We have setup and currently have a Buildbot process performing tests not only on the UNIC code and its components, but also the legacy tools such as MC²-3. Although we do not have sufficient test problems for MC²-3 at this point, nor the inline cross section generation work, these will be incorporated as the development proceeds. To handle the export issues, MC²-3 will continue to be distributed through RSICC [21] and each working version of UNIC will be exported to the SHARP repository for use by the SHARP collaborators. Eventually we conceive of releasing UNIC through RSICC, but we do not currently have enough funding to support exporting a production quality tool.

6 Verification and Validation Tests

Consistent with the preceding years, a significant amount of time was spent in FY2010 benchmarking the neutronics package of UNIC. This work still focuses primarily on fast

reactor technology given that the thermal reactor cross section processing involves a procedure much more complicated than the fast reactor procedure. Thus, we still expect a few years of development will be needed before a generic reactor modeling capability is ready within UNIC.

6.1 Follow on Calculations of ZPR-6/6A

The ZPR-6 Assembly 6A problem was pursued in FY2009 to prove that the second order methodology could be used on large heterogeneous benchmark problems. While we ideally would obtain more accurate solutions using the heterogeneous modeling than the homogeneous modeling, the inability to fully resolve the problem in space, angle, and energy prevents this from occurring. As a consequence, the focus on ZPR-6/6A was to understand the source of errors in the preceding year and try to do a better job on defining the input geometry and cross section data.

One problem observed in the FY2009 work was an inaccurate eigenvalue result on the drawer homogenized model which we identified as an error in the anisotropic scattering data inputted into UNIC. Table 6.1 gives the revised eigenvalue solutions which demonstrate the expected eigenvalue comparison to the reference VIM solution for the as built experimental configuration. In addition to resolving this issue, the simplified ZPR-6/6A problem was used to further investigate the scalability of the SN2ND solver. In the past couple of years the SN2ND solver methodology has been tested on several high performance computing machines. The previous weak angle scaling study carried out on BlueGene/P at Argonne National Laboratory was extended to 294,912 cores by using the Jugene super computer [24], which is essentially a larger version of Argonne’s BlueGene/P. Table 6.2 shows the updated weak scaling in angle and shows that SN2ND can achieve 76% scaling performance on a machine with the largest processor count in the world.

Table 6.1. SN2ND Eigenvalue Error for the Drawer Homogenized ZPR Model

Energy Groups	Eigenvalue Error (pcm)
9	26
33	-15
116	-16
230	-15
VIM	0.99981 ±0.00025

Table 6.2. Weak Angle Scalability of SN2ND on BlueGene/P (combined ANL and JSC)

Total Cores	4π Angles	Total Time (sec)	Weak Scaling
32,768	32	579	100%
73,728	72	572	101%
131,072	128	581	100%
163,840	160	691	84%
294,912	288	763	76%

The simplified model was created because of the complicated geometry and meshing associated with being able to construct the input for the “as built” configuration. As stated, the consequence of this was further modification of the BuildZPRmodel code in FY2010 to better accommodate the needs of modeling a ZPR experiment. Before that work was finished, an attempt was made to model the as built configuration by manually setting up input for each individual drawer; a one month process. Figure 6.1 shows the flux and power solution such that the difficulties modeling the axial plate geometry, interleaved plates, are well displayed.

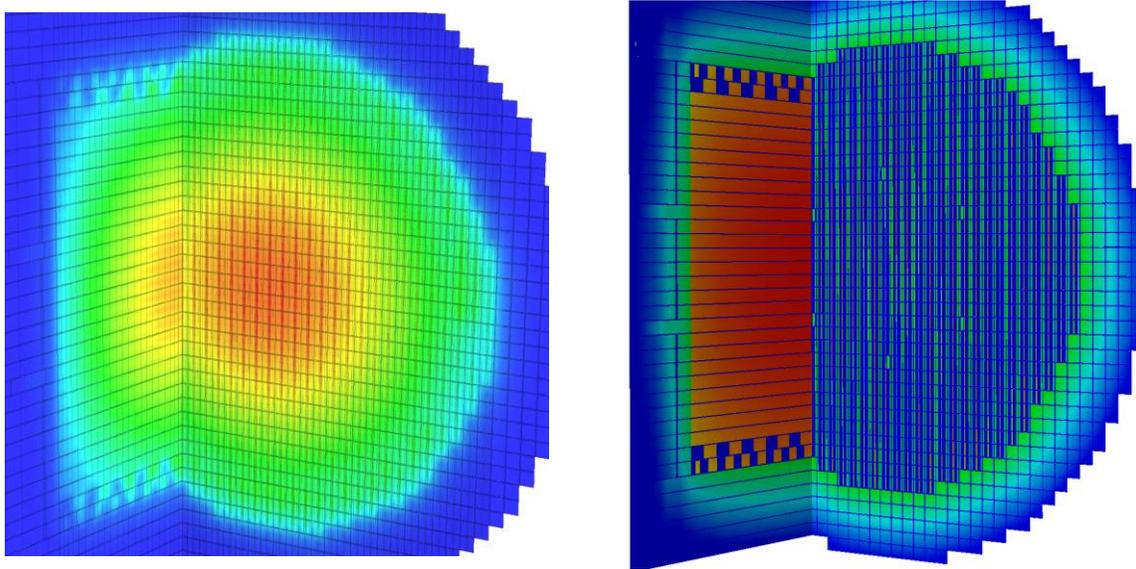


Figure 6.1. ZPR-6/6A Fast Flux (left) and Power (right)

Using this input geometry, a new study was carried out on the XT5 machine [23] where we used a maximum of 222,912 cores and over 0.6 trillion degrees of freedom. The intent was to assess the accuracy of the ZPR-6/6A model using the best possible space-angle-energy resolution in SN2ND. The additional fact that the geometry and cross section mapping were done by hand necessitated using a minimal (potentially erroneous) cross section generation procedure. With respect to the resolution, a maximum of 116 groups, a mesh with 50 million vertices, and a cubature with 432 angles were tested although not all are combined in one calculation. Table 6.3 shows the convergence behavior of the eigenvalue of SN2ND for two different meshes where 32 angles was used at the coarser mesh and 100 angles was used on the finer mesh, and Table 6.4 shows the eigenvalue convergence behavior with respect to angle using the 33 group cross section set.

Table 6.3. SN2ND Eigenvalue Error (pcm) Energy Convergence for Two Different Meshes

Mesh Vertices Millions	9 group	33 group	116 group
9.6	-761	-918	-973
19.6	-709	-855	

Table 6.4. SN2ND Eigenvalue Error (pcm) Angular Convergence

Angles	9.6 million	19.6 million
32	-937	-715
72	-969	-752
128	-951	-738
200	-919	-709
288	-888	
392	-860	
432	-814	

From Table 6.3, the change in energy yields significantly larger errors than that of the drawer homogenized model results in Table 6.1. Neither table shows a clear path to convergence with the angular variable clearly not converged in Table 6.4. A good mesh refinement study was not possible (we executed a single calculation with the 50,000,000 vertex mesh) because of insufficient computing power (or lack of a good multigrid preconditioner). However, from the results displayed it is highly likely that mesh convergence has not been achieved.

Although it is clear how much remaining error is present in the eigenvalue, we cannot get a sense of how much is derived from space, angle, and energy refinement. We must also concede that there can be errors resulting from the manual geometry creation, the simplified cross section generation methodology, and the fact that plate-by-plate cross section data may be completely unreliable at this point. From any viewpoint, this calculation gives a sobering assessment of the space-angle-energy requirements of an explicit geometry representation of a real reactor. Even if the above turns out to be an erroneous input specification, it is not clear whether detailed heterogeneous modeling of nuclear reactor calculations can be considered practical on the best HPC machines. While this was not unexpected, it is a motivation for building simplified neutronics modeling tools such as those studied in the NNR and NODAL as part of the SHARP effort. As a final note, similar to the BlueGene/P results, SN2ND displayed 75% weak scaling on 222,912 cores of XT5 and nearly identical strong scaling performance to that observed on BlueGene/P (+90%).

6.2 Space-angle Convergence Study of PN2ND and SN2ND

In the previous years we have observed several test problems with residual eigenvalue errors when compared with a comparable multi-group Monte Carlo reference solution (not obtained directly by ANL). One specific case was the Takeda benchmark series upon which rigorous mesh and angle refinement were required to resolve the eigenvalue error. To ensure that these errors are not systematic, additional work was carried out this year where composition and 9 group cross section data derived from the MONJU reactor was combined with the test geometries shown in Figure 6.2 in an attempt to simulate the problem experienced with the Takeda benchmarks.

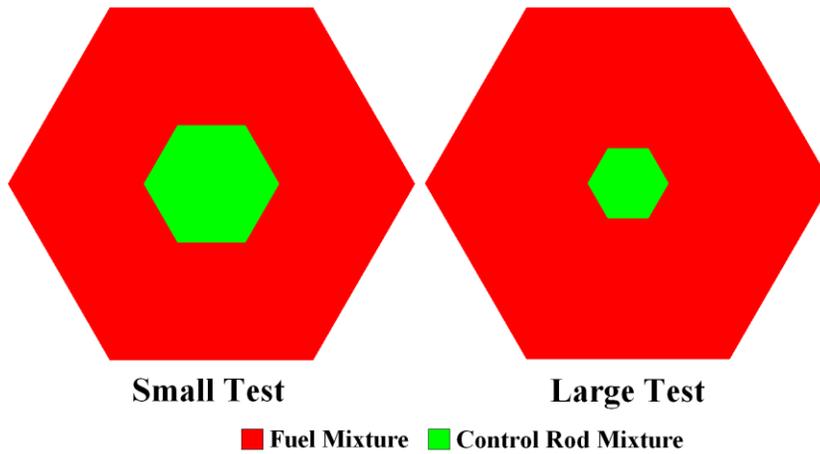


Figure 6.2. Space-angle Mesh Convergence Study Test Problem

In both test calculations, the inner hexagon is the same dimension of 11.594 cm and thus the large test has a larger fuel region around it. Table 6.5 gives the PN2ND eigenvalue results for several space-angle approximations on the small test problem while Table 6.6 and Table 6.7 give the eigenvalue results for the large test problem with reflected and vacuum boundary conditions, respectively, where the reference solution is taken as the highest space-angle solution obtained for each problem.

Table 6.5. PN2ND Results for the Small Test Model with Reflective Boundary Conditions

P _N Order	6 Tri/Assembly	24 Tri/Assembly	54 Tri/Assembly	96 Tri/Assembly
3			- 0.00060	- 0.00058
5		-0.00021	- 0.00015	- 0.00013
7	- 0.00062	- 0.00014	- 0.00006	- 0.00004
9	- 0.00061	- 0.00012	- 0.00003	- 0.00001
11	- 0.00061	- 0.00011	- 0.00002	1.04134

Table 6.6. PN2ND Results for the Large Test Model with Reflective Boundary Conditions

P _N Order	6 Tri/Assembly	24 Tri/Assembly	54 Tri/Assembly	96 Tri/Assembly
1			- 0.00065	- 0.00065
3		-0.00022	- 0.00020	- 0.00019
5	- 0.00025	- 0.00008	- 0.00005	- 0.00005
7	- 0.00023	- 0.00005	- 0.00002	- 0.00001
9	- 0.00023	- 0.00005	- 0.00001	1.21363

Table 6.7. PN2ND Results for the Large Test Model with Vacuum Boundary Conditions

P _N Order	6 Tri/Assembly	24 Tri/Assembly	54 Tri/Assembly	96 Tri/Assembly
3				- 0.00093
5			- 0.00076	- 0.00025
7		- 0.00016	- 0.00011	- 0.00010
9		- 0.00011	- 0.00005	- 0.00003
11	- 0.00046	- 0.00009	- 0.00002	0.53573

For PN2ND, nearly full convergence is observed on three problems using a P_{11} angular approximation and 96 quadratic prismatic elements per assembly. The Monte Carlo multigroup eigenvalue for the small problem is 1.04129 ± 0.00011 and is 1.21362 ± 0.00012 for the large problem with reflected boundary conditions and 0.53587 ± 0.00011 with vacuum boundary conditions. In all three cases the PN2ND solver solutions are within the statistical error on the Monte Carlo solution and further space-angle mesh refinement is unnecessary. The comparable SN2ND solutions exhibit nearly identical results. From these results we see the typical convergence behavior observed on all of the homogenized fast reactor problems UNIC has been applied to thus far (ABTR, MONJU, ZPR, etc.) not only accounting for these 9 group calculations, but also 33, 70, 116, and 230 group models. As a result, we can only conclude that the cross sections generated for the Takeda 4 benchmark are not likely to appear again for the fast reactor problems in the near future.

6.3 ZPR-6 Assembly 7 Experiment

The ZPR-6 Assembly 7 benchmark [25] was carried out immediately after the ZPR-6 Assembly 6A experiment with the intention of studying plutonium core rather than uranium. Unlike the ZPR-6/6A work of last year, we were unable to use SN2ND on a full core heterogeneous model. While we were able to build the geometry with BuildZPRmodel, thereby greatly improving the modeling accuracy over the ZPR-6/6A model, we did not get enough large scale runs through the queue on BlueGene/P to provide any usable information at the time of this report. Because of the lower computing resource requirements, we were able to get the homogenized drawer models of the experiment executed and evaluated for four loadings of the ZPR-6/7 such that we compared UNIC results to foil measurements from the experiments.

6.3.1 Slab Geometry Scoping Studies

Rather than directly start solving the full core ZPR calculations we choose to do some studies on the slab geometry model to see how the cross section generation procedure impacts the accuracy of the result. The typical core drawer model for ZPR-6/7 is shown in Figure 6.3.

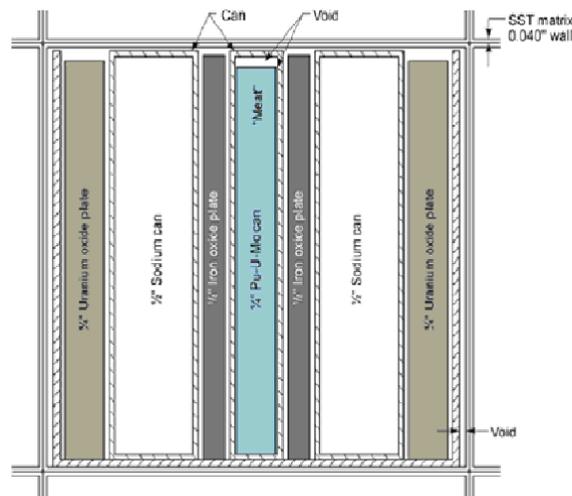


Figure 6.3. Representative ZPR-6 Assembly 7 Core Drawer Layout

Given the BuildZPRmodel code in hand, the concept behind this work was to determine the best possible scheme to use inside of BuildZPRmodel for generating the slab geometry MC²-3 input data. The overriding goal of course is to do the best possible job of using UNIC and MC²-3 in calculating the foil reaction rates measured during the experiment.

The first test considered the impact of smearing the plate masses guided by previous experience [26-27]. In the first approach, termed Model A, the plate dimensions and heavy metal compositions in the X direction are preserved with the extra (Y & Z directional) structural material added to the non-heavy metal plates according to their respective thicknesses. This model was used in the energy self-shielding part of MC²-3 while the geometry was modified in the spectrum calculation such that the heavy metal plate compositions are diluted to preserve the overall material balance in the three-dimensional to one-dimensional representative transformation. In the second approach, termed Model B, the modified geometry case was used in both calculation steps of MC²-3.

The second test considered the same schemes but, during the generation of the one-dimensional model, the impact of smearing steel in with the sodium (option 2) as opposed to the other steel materials (option 1) was studied. Table 6.8 shows the eigenvalue results for the combined set of tests for the primary core drawers of ZPR-6 Assembly 7. As can be seen in Table 6.8, the various alterations to the one-dimensional model have a relatively minor impact on the MC²-3 solution and there is a consistent deviation of ~200 pcm in the multiplication factor between 3-D and 1-D models. However, as discussed below, the 1D MC²-3 models appear to be adequate for generating mutigroup cross sections for full core calculations as far as the final group structure is sufficiently fine.

Table 6.8. MC²-3 k_{∞} results for a Single Fuel Drawer of the ZPR-6/7, Loading 104

Drawer Master 577 (Representative High Pu240 Core Drawer)					
3-D MCNP Monte Carlo $k_{\text{eff}} = 1.28313 \pm 0.00028$					
1-D Monte Carlo		1-D MC ² -3			
Option 1	Option 2	Option 1		Option 2	
		Model A	Model B	Model A	Model B
1.28115 ± 0.00017	1.28167 ± 0.00020	1.28445	1.28396	1.28451	1.28405
Drawer Master 719 (Representative Normal Core Drawer)					
3-D MCNP Monte Carlo $k_{\text{eff}} = 1.25742 \pm 0.00027$					
1-D Monte Carlo		1-D MC ² -3			
Option 1	Option 2	Option 1		Option 2	
		Model A	Model B	Model A	Model B
1.25512 ± 0.00019	1.25527 ± 0.00018	1.25877	1.25824	1.25859	1.25807

6.3.2 Eigenvalue Results for the Homogenized Drawer Calculations

The high Pu-240 cores of ZPR-6 Assembly 7 were set up and executed for the Loadings 104, 106, 120, and 132. The intent was to see how well MC²-3 combined with SN2ND could perform for predicting the core reactivity and reaction rate distributions. Figure 6.4 shows the material configuration for all four loadings while Figure 6.5 shows some selected flux plots from the 70 group calculation of Loading 106. The specific details of each loading are detailed elsewhere [25,28] and are not reproduced here. It is noted that in the Loadings 106 and 132 shown in Figure 6.4, four BeO drawers are loaded into the center of the core. These

BeO drawers were introduced to enhance the reactivity worth of the B₄C simulated control rod that appears in the Loading 132. These drawers cause significant localized flux changes compared with drawers that do not contain the beryllium as evident in Figure 6.5.

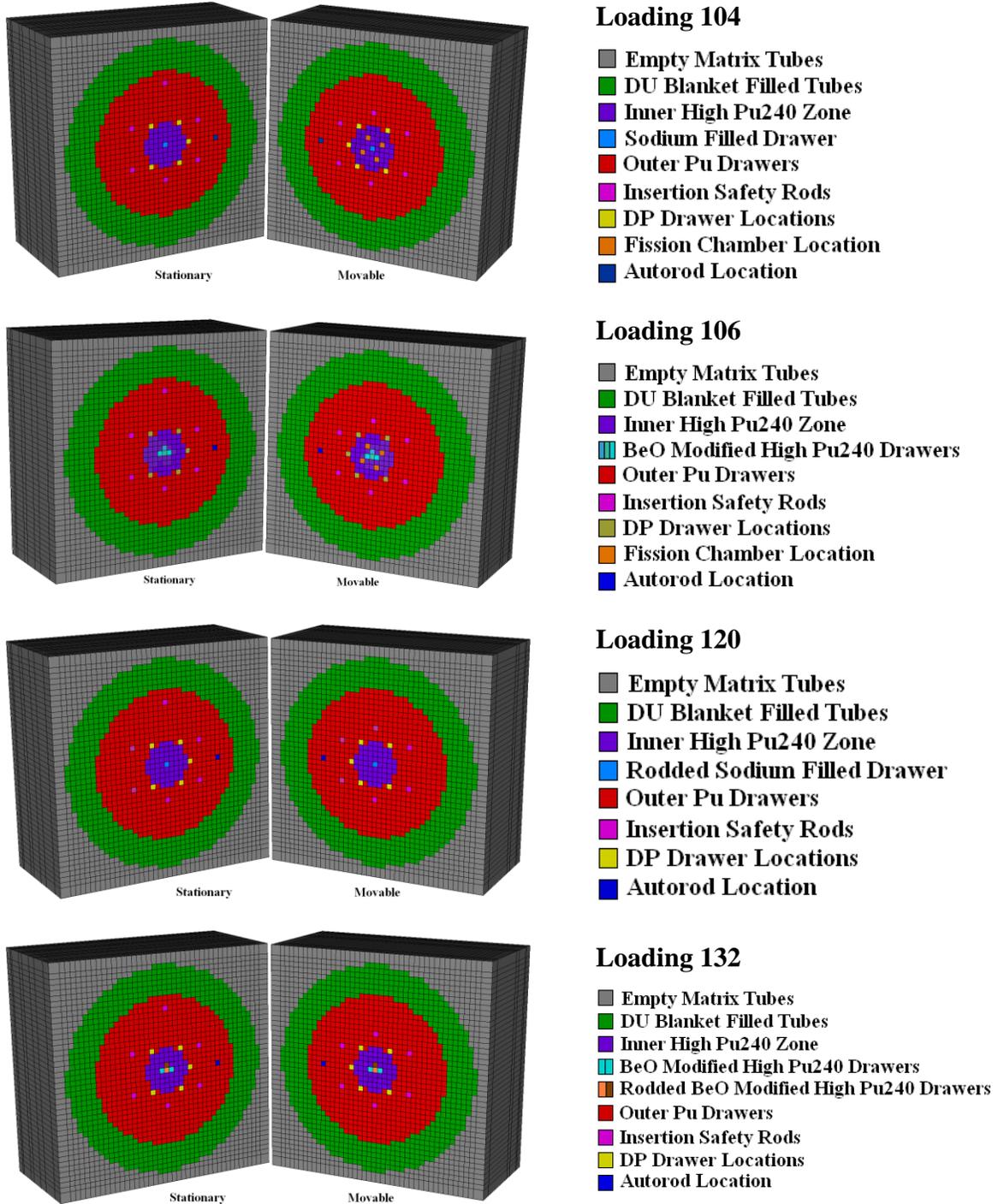


Figure 6.4. ZPR-6 Assembly 7 Loadings 104, 106, 120 & 132

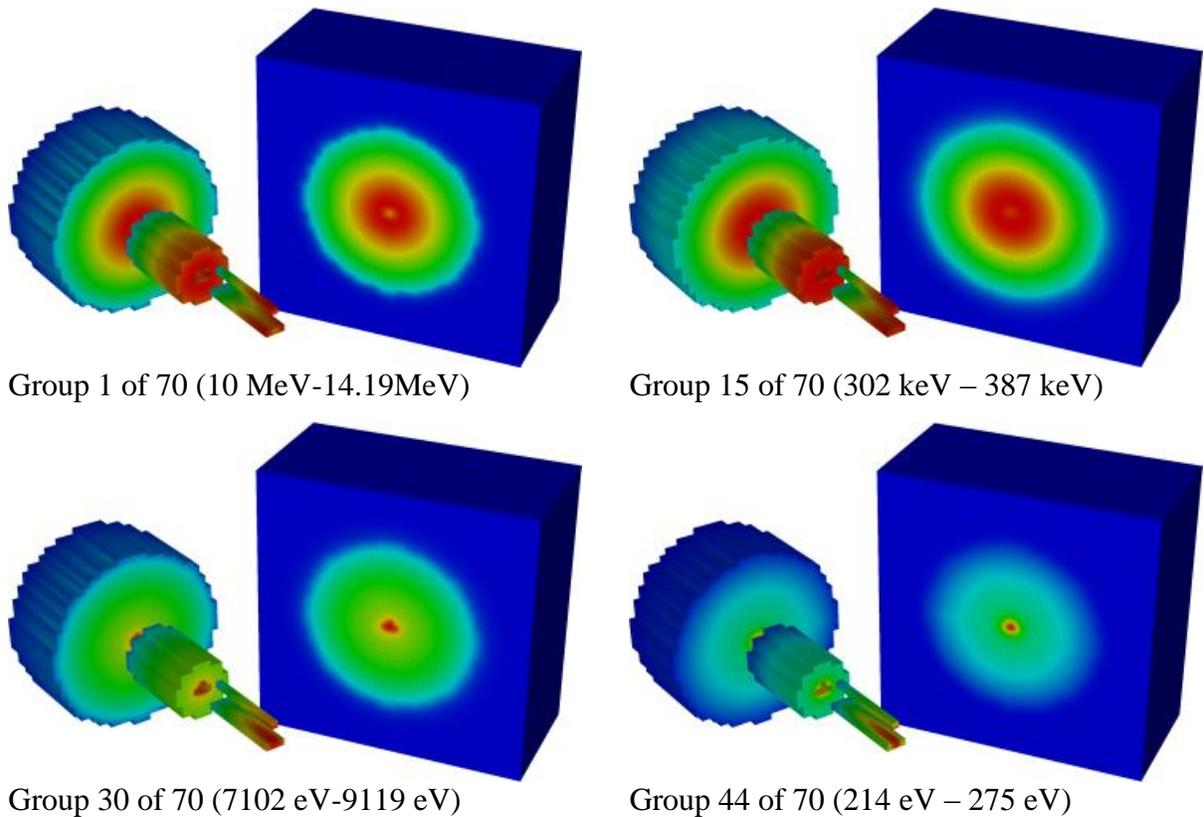


Figure 6.5. ZPR-6 Assembly 7 Selected Flux Plots for Loading 106

For each plot in Figure 6.5, the right hand picture displays the face of the movable matrix half as viewed in Figure 6.4 while the left hand picture shows the flux solution for the active core portion of the stationary side (everything inside of the blankets). For the stationary side, we separated the high Pu-240 zone to display the axial flux solution along with the central BeO modified drawers such the axial flux variation can be observed. As can be seen, there is only a modest impact of having the BeO drawers at the higher energy levels and a modest impact on the unresolved resonance region. In the epithermal ranges we can observe a substantial peak in the flux solution due to the improved scattering source derived from the BeO drawers. Combined, these strong lower energy peaks are not properly handled by the one-dimensional single drawer model and are thus believed to cause various errors in the local reaction rates near the BeO plates.

Table 6.9 summarizes the eigenvalue results for loading 104 which displays convergence with respect to energy, mesh, and angular cubature, along with the importance of anisotropic scattering. We also included the results obtained using PN2ND in diffusion theory (uses the transport cross sections provided by MC²-3).

Table 6.9. Eigenvalue Solutions for ZPR-6 Assembly 7 Loading 104

Mesh Vertices	Angular Cubature	Scat. Order	Number of Energy Groups				
			9	33	70	116	230
57,132	L3T3	P1	0.99676	0.99828	0.99905	0.99880	0.99905
223,928	L3T3	P1	0.99845	1.00001	1.00080	1.00055	1.00081
805,185	L3T3	P1	0.99854	1.00010	1.00089	1.00064	1.00090
805,185	L3T3	P3	0.99888	1.00044	1.00123		
805,185	L5T5	P1	0.99853	1.00009	1.00088	1.00063	1.00089
805,185	L5T5	P3	0.99887	1.00044	1.00122		
1,467,429	L3T3	P1	0.99855	1.00011	1.00090	1.00065	1.00092
1,467,429	Diffusion	P0	0.99911	1.00065	1.00145	1.00105	1.00130

Hexahedral meshes were used for all of these calculations where the 57,132 vertex mesh is considered a test case since it uses linear hexahedral trial functions. The 223,928 vertex mesh assumes ~8 cm axial element sizes and one hexahedron per homogenized drawer. The 805,185 vertex mesh also uses ~8 cm axial element sizes but it defines 4 hexahedrons per homogenized drawer which yields about a 9 pcm change in the eigenvalue compared with the 223,928 vertex mesh. The 1,467,429 vertex mesh also uses 4 hexahedrons per homogenized drawer but it has a 5 cm axial mesh size which yields no significant improvement over the 805,185 vertex mesh. In general, we can consider the 805,185 vertex mesh to be sufficiently mesh refined for our needs.

The square Legendre-Tchebychev angular cubature was chosen for all of these calculations where L3T3 is roughly equivalent to a standard level symmetric S_4 cubature (32 versus 24 directions) and L5T5 is roughly equivalent to a S_6 cubature (72 versus 48 directions). From Table 6.9, there is virtually no impact of the angular cubature on the eigenvalue solution of this homogenized drawer problem, but there is ~40 pcm error associated with anisotropic scattering which is quite significant. We did not attempt using P_5 anisotropic scattering, but experience typically indicates that there is not much difference between P_3 and P_5 on these homogenized benchmark problems. The convergence with respect to energy is quite curious because the 70 group and 230 group solutions appear identical while the 116 group result would indicate that the 70 group solution should be less accurate. This might have something to do with the fact that the 70 group structure was optimized for fast reactor problems such as Loading 104 while the 116 and 230 group are simple equal lethargy approaches.

Table 6.10 compares the 70-group eigenvalue results for all four loadings 104, 106, 120 and 132 obtained with the most refined mesh and angle settings and P_3 scattering with the MCNP solutions for the as-built core models. The measured reactivity values are also shown with the measurement uncertainty; the combined uncertainty due to geometry and composition uncertainties estimated for the Loading 99 was ~80 pcm. Note that we have been unable to execute the P_3 116-group or 230-group calculations using the SN2ND solver due to memory limitations on BlueGene/P.

Table 6.10. UNIC and MCNP Eigenvalues for Loadings 104, 106, 120 and 132

	Loading 104	Loading 106	Loading 120	Loading 132
UNIC	1.00147	1.00134	1.00127	1.00016
MCNP	1.00016 ±0.00007	1.00049 ±0.00007	0.99967 ±0.00007	1.00040 ±0.00007
Measurement	1.00072 ±0.00002	1.00091 ±0.00003	1.00099 ±0.00003	1.00040 ±0.00002

* Measurement uncertainty only, excluding composition and geometry uncertainties

As can be seen, the deterministic SN2ND results are generally higher than the MCNP calculations for all but the loading 132 calculation. However, it is noted that the MCNP solutions generally underestimate the core reactivity compared to the measurements while the UNIC solutions overestimate them. For all the four core loadings analyzed, the UNIC predicted the core reactivity within 1- σ (standard deviation) of the estimated experimental uncertainty (~80 pcm). This result is comparable to the accuracy of MCNP solutions; the UNIC solutions deviated from the measured values by 75, 43, 28 and -24 pcm for the Loadings 104, 106, 120, and 132, respectively, while the corresponding deviations of MCNP solutions were -56, -42, -132, and 0 pcm.

6.3.3 Foil Results for the Homogenized Drawer Calculations

As mentioned, the goal for this year was to compare the UNIC+MC²-3 calculations directly with the experimental foil measurements. The foil measurements were analyzed for the four loadings, but we discuss in this report only the results for the Loadings 104 and 106, since the results for the Loadings 120 and 132 were very similar to those of the Loadings 104 and 106, respectively.

Four types of foils were used to measure relative reaction rates: enriched uranium (EU) foils to measure ²³⁵U fission rates; depleted uranium foils to measure ²³⁸U capture and fission rates; and two types of plutonium (Pu) foils to measure ²³⁹Pu fission. These foils were placed in 2-in. square packets in the fronts of the drawers of the stationary half between the Fe₂O₃ plate and the fuel plate on the left side of the drawer (see Figure 6.3) as one faces the front of the drawer from a position between the halves. Figure 6.6 shows the positions of the foils within the packets and the positions of the packets in the core for the foil irradiations. Each packet consisted of the four foil types sandwiched between 1-mil aluminum and 1-mil stainless steel squares on both sides of the packet.

For the foil reaction rate calculations, the flux values of each foil were determined by superimposing the plate-wise flux shapes from MC²-3 onto the global flux solution from UNIC. The reaction rates were calculated by combining these flux values with the 230-group homogenized drawer cross sections of MC²-3. The calculated results are compared with the experimental measurements in Table 6.11. As can be seen, the results for the Loading 104 are in good agreement with the measurements, considering the experimental uncertainties: 1.6% for EU fission, 2% for DU capture, 2.8% for DU fission, and 1.5% for Pu fission. However, it is noted that the EU fission and DU capture rates of the Loading 106 show significant errors near the BeO drawer locations (i.e. small Y values).

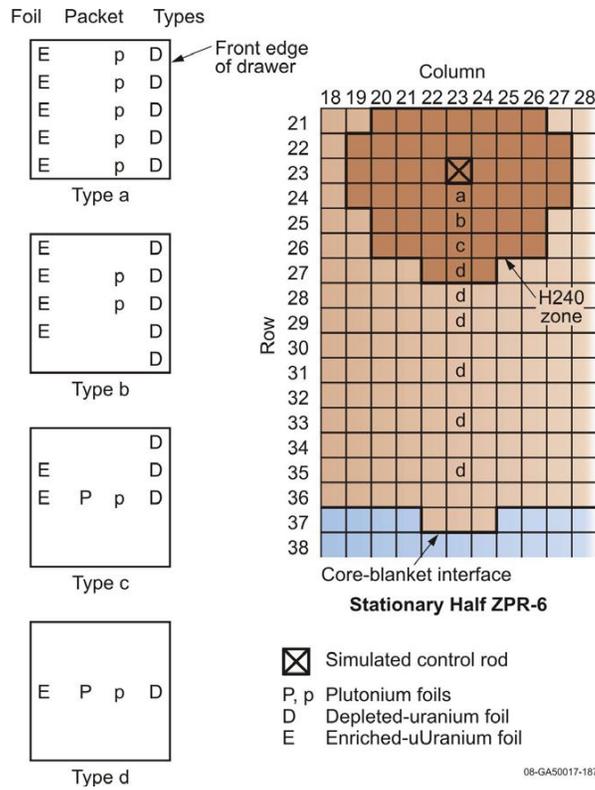


Figure 6.6. Foil Locations in the ZPR-6 Assembly 7 High ²⁴⁰Pu Loadings

Table 6.11. Foil Reaction Rate Comparisons for Loading 104 and 106 (C/E-1 in %)

Y (cm)	Loading 104				Loading 106			
	EU (f)	DU (c)	DU (f)	Pu (f)	EU (f)	DU (c)	DU (f)	Pu (f)
3.61	2.34	-1.21	2.42	2.85	-4.58	-16.61	1.93	-1.96
4.25	2.61	0.40	0.50	3.43	-1.31	-9.95	-1.23	0.95
5.52	2.67	1.45	0.40	3.03	-0.55	-6.03	-0.79	1.38
6.79	2.67	1.47	1.36	2.93	-0.34	-4.69	-0.10	1.30
7.42	2.58	0.32	2.28	2.43	-0.17	-4.05	0.96	0.69
9.13	2.16	0.28	3.08		0.22	-2.44	2.29	
9.77	2.40	1.35	1.89	2.49	0.81	-2.38	1.24	1.78
11.04	2.27	1.69	1.31	2.07	0.92	-0.77	1.07	0.97
12.31	1.86	1.17	1.83		0.30	-1.22	1.16	
12.94		0.45	3.28			-1.07	2.64	
14.65		-1.43	3.48			-1.62	2.98	
15.29	1.48	0.85	2.29		0.53	-0.32	1.69	
16.56	1.59	1.21	1.63	1.62	0.57	0.17	1.05	0.41
22.08	1.93	1.72	1.75	2.31	0.60	0.83	1.70	0.66
27.6	1.98	0.63	3.19	2.21	1.31	0.31	2.81	1.32
33.12	1.99	0.27	2.70	2.15	1.73	1.16	2.52	1.02
44.16	1.56	-1.07	2.39	1.69	0.81	-0.52	1.63	1.01
55.19	1.79	0.56	1.51	0.99	1.19	0.91	0.89	1.08
66.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

The presence of the BeO drawers introduces a strong tail in the epithermal neutron flux as displayed in Figure 6.5. The calculated foil reaction rates in Table 6.11 are not satisfactory, and we investigated whether space-angle-energy refinement in UNIC was the source of error. Comparison of predicted foil measurements for several different mesh, angle, and energy structures revealed only a minor dependence (second significant digit) on the predicted foil measurements. As a consequence, we are left with the cross section generation procedure which uses drawer-wise slab lattice calculations in MC²-3. Basically, the single drawer MC²-3 model with reflective boundary conditions is not able to account for the spectral interactions between the fuel drawer and the BeO drawer, and thus introduces significant errors in the cross sections for the resolved resonance range. The ideal way to handle this in MC²-3 is to utilize a two-dimensional (or three-dimensional) model which we were unable to complete this year because of unanticipated delays in the MOCFE solver development.

6.3.4 Follow on Work to Investigate the BeO Related Errors

Because UNIC has a general unstructured mesh capability, the drawer homogenized model of the Loading 106 was modified to include the BeO plates heterogeneously in the domain. The purpose was to see how much of the error was caused by smearing the BeO plates with the fuel and other plates in the drawer. The updated reaction rate predictions are shown in Table 6.12.

Table 6.12. Foil Reaction Rate Comparisons for Loading 106 with Explicit BeO Plates

Y (cm)	Heterogeneous BeO Ring			
	EU (f)	DU (c)	DU (f)	Pu (f)
3.61	3.46	-11.12	-0.79	-0.32
4.25	5.46	-5.29	-3.19	2.32
5.52	4.26	-2.73	-2.17	2.37
6.79	3.24	-2.30	-1.65	2.10
7.42	2.88	-2.04	-0.59	1.42
9.13	2.18	-1.22	1.09	
9.77	2.47	-1.39	0.24	2.44
11.04	2.12	-0.11	0.46	1.68
12.31	1.14	-0.81	0.86	
12.94		-0.75	2.45	
14.65		-1.49	2.93	
15.29	0.87	-0.24	1.66	
16.56	0.79	0.17	1.02	1.29
22.08	0.50	0.65	1.45	1.49
27.60	1.13	0.08	2.46	1.92
33.12	1.54	0.94	2.19	1.56
44.16	0.67	-0.66	1.35	1.43
55.19	1.13	0.85	0.70	1.28
66.23	0.00	0.00	0.00	0.00

Comparison of Table 6.11 to Table 6.12 shows a marginal improvement in the Loading 106 foil measurement predictions. Here we note that in addition to the higher U-238 capture rates, there is an obvious increase in U-235 fission rates, both of which confirm the suspected spectral shift to a softer spectrum near the core central region.

Continuing, we also considered errors derived from the foil cross sections themselves. For example, instead of using drawer homogenized cross sections for U-235, U-238 and Pu-239 to calculate the foil reaction rates, MC²-3 was used to generate infinite dilute cross sections for all relevant isotopes and flux peaking factors at foil locations inside the slab geometry models. These were used with UNIC solutions for homogenized drawer models to calculate foil reaction rates through a flux reconstruction procedure. In addition, another model in which all foil types were homogenized inside the 2x2 inch foil packet was used to generate the necessary cross sections and flux peaking factors in an explicit slab model of the foils. Again, foil reaction rates were reconstructed using UNIC fluxes. The results for U-238 capture for these two cases are summarized in Table 6.13.

Table 6.13. U-238 Capture Foil Reaction Rate Comparisons for Loading 106 with Modified Foil Cross Section Models

Y (cm)	Foil Cross Sections Model		
	Homogenized Drawer Cross Sections	Infinite Dilute Cross Sections	All Foils Homogenized 1-D Drawer Model
3.61	-11.12	33.25	5.97
4.25	-5.29	35.50	10.01
5.52	-2.73	30.00	3.46
6.79	-2.30	24.08	2.49
7.42	-2.04	21.56	2.08
9.13	-1.22	16.26	1.78
9.77	-1.39	14.15	1.11
11.04	-0.11	12.43	1.89
12.31	-0.81	9.14	0.33
12.94	-0.75	8.17	0.29
14.65	-1.49	5.21	-0.38
15.29	-0.24	5.87	1.08
16.56	0.17	5.21	0.92
22.08	0.65	3.15	1.56
27.6	0.08	2.75	1.44
33.12	0.94	3.35	1.85
44.16	-0.66	1.62	0.65
55.19	0.85	3.04	2.07
66.23	0.00	0.00	0.00

Because infinite dilute cross sections do not account for any resonance self-shielding effects, U-238 capture is obviously over predicted and we obtain much worse results. On the other hand, when the foils were homogenized together inside the foil packet we see an improved comparison with the experimental results except for the first two positions closest to the modified BeO drawers. However, it should be noted that the foil homogenization strategy is somewhat arbitrary.

As mentioned, all of this work was done to determine the sensitivity of the foil reaction rates with regard to the cross section process algorithm. In most cases we observed relatively minor dependence on the space-angle-energy approximation incorporated in UNIC and a minor but important dependence on the method used to obtain the foil cross sections. In the coming year we hope to implement the MOCFE solver capability within MC²-3 such that we can further research the source of these errors and thereby at least improve the existing homogenization methodologies.

7 Conclusions

A considerable amount of work was done on UNIC and the supporting tools surrounding it. Specifically, we investigated large scale heterogeneous calculations using the SN2ND solver and started the work required to build a multigrid preconditioner. Additional development time was spent on the MOCFE solver to resolve outstanding issues with the parallel algorithm along with improving the reliability of the solver. Finally, we spent a considerable amount of time creating a new tool, BuildZPRmodel, to assist in the analysis work being carried out on the ZPR experiments. With these tools, we performed detailed calculations on the ZPR-6/6A and ZPR-6/7 experiments, providing comparisons of predicted versus experimental foil activation on the latter.

Additional studies using SN2ND on ZPR-6/6A indicated 76% weak scaling in angle on 294,912 cores of the BlueGene/P machine and 75% weak scaling in angle on 222,912 cores of XT5. In the previous year we demonstrated strong scaling in space (without multigrid) at 80% or higher and strong scaling in angle at ~70%. While we could not fully resolve the ZPR-6/6A problem, the general conclusion is that the methodology of SN2ND can legitimately be used to solve heterogeneous problems such as ZPR and other comparable fast reactor problems. As a consequence, a new revision of the SN2ND was started this year to accommodate a multigrid preconditioner which is the only way to achieve good performance on the large spatial meshes needed for the heterogeneous problems. While the space-angle-energy requirements of such calculations do not appear to be practical on the best HPC machines today, the next generation machine might prove to have the necessary computing capability to facilitate such heterogeneous calculations. While this was not unexpected, it is a motivation for building simplified neutronics modeling tools as part of the SHARP effort such as NODAL and those studied in the NNR. With these tools we can not only support the routine research needs of existing reactor analysis, but we can also explore the high end modeling capabilities that HPC machines offer.

From the preceding discussion on SN2ND, it should be clear that not all goals set out for FY2010 were accomplished. Most important of these of course is the multigrid preconditioner capability of the SN2ND such that UNIC can facilitate solving large heterogeneous problems such as ZPR-6/6A and ZPR-6/7. While we believe that we have identified a viable multigrid strategy using linear tessellation followed by algebraic multigrid and have setup the

appropriate mapping functions within UNIC, we were unable to fully implement the scheme in a usable version of SN2ND this year. Thus we can expect to dedicate some time to that work in the next year.

The MOCFE solver does not have as clear of a path to scalability as the SN2ND solver. As an example, a multigrid preconditioner in SN2ND is straightforward because the system of equations can be progressively broken down into energy, angle, and space. With respect to space, SN2ND requires a preconditioner which is diffusive-like and very amenable to conventional multigrid techniques, although we note that very few specific multigrid applications have been applied to date on the SN2ND related system. With MOCFE, we do not have a clear multigrid strategy and, to be honest, we do not even have a good starting point for a preconditioner. Combining this with the fact that the coefficient matrix application ($A \cdot x$) is not easily scalable, as discussed and demonstrated in this report, the MOCFE development path is much more difficult than SN2ND.

Nevertheless, in FY2009 we focused a considerable amount of effort rebuilding the initial MOCFE solver into a form which obeys the basic requirements of a scalable algorithm: memory and communication per process must scale. In FY2010 we identified and resolved several issues with parallel ray tracing and developed a scalable back projection methodology. As discussed, the three-dimensional back projection algorithm is an approximation to the true system and we were unable to test out the approach with respect to accuracy or scalability. Overall, the FY2010 research on MOCFE indicates that there is a considerable amount of work left to ensure a load balanced coefficient matrix-vector operation in addition to a memory tolerable ray tracing algorithm which will continue into the next year. Assuming that these tasks are fixable, we then have the arduous task of identifying and implementing an appropriate preconditioner for the MOCFE solver.

The NNR project was merged in with the SHARP project this year thus necessitating some rethinking of the project goals of both. Both approaches have good ideas with regard to performing reactor analysis on the respective targeted reactor types. To partially fulfill the need to carry out fast reactor analysis using existing methodologies, we spent some time this year developing NODAL. The primary focus was to identify an acceptable preconditioner for the resulting discretized equations which we believe we accomplished. In the coming years we can expect to incorporate the best parts of both SHARP and NNR and create a set of higher accuracy tools which can meet both thermal and fast reactor analysis needs.

The BuildZPRmodel tool started in the previous year required a considerable amount of follow-on work because of the ZPR-6/7 related studies. The primary enhancements are the ability to create input files for cross section generation and connecting the cross section data with the materials used in the homogenous model (whether it is drawer-wise or plate-by-plate). This tool was exclusively used to create the input for the drawer-wise homogenized SN2ND calculations of ZPR-6/7 and the associated foil activation comparisons. The addition of a "solution along a line" analysis capability greatly simplified that effort as discussed in the report. BuildZPRmodel was also used to create a plate-by-plate model of ZPR-6/7 and some test calculations were performed, but there are insufficient results to warrant inclusion in this report. We fully expect to revisit both ZPR-6/7 and ZPR-6/6A in the coming years with any new improvements to SN2ND and MC²-3 to try and resolve the outstanding issues with both accuracy and performance discussed in this report.

The most important work done this year was the ZPR-6/7 benchmarking work. Unlike the previous years, we focused a considerable amount of effort this year to not only verify the eigenvalue answer, but also the flux solution using the foil measurements taken during the experiment. We carried these calculations out using drawer-wise homogenized models of ZPR-6/7 and, as part of a scoping study, a partially dehomogenized model (explicit BeO plates). For the ZPR experiments that have conventional fast reactor materials (i.e. Loadings 104 and 120), we observed rather accurate solutions using UNIC. For all the four core loadings analyzed, the core reactivity was predicted within $1\text{-}\sigma$ (standard deviation) of the estimated experimental uncertainty (~ 80 pcm), including the geometry and composition uncertainties. This result is comparable to the accuracy of MCNP Monte Carlo solutions. For the Loadings 104 and 120 of conventional fast reactor compositions, the calculated reaction rate distributions agreed well with the foil activation measurements within 1- to $2\text{-}\sigma$ of the measurement uncertainties. However, for the Loadings 106 and 132 containing BeO plates, the cross section generation methodology proved to be insufficient and thus more than $3\text{-}\sigma$ deviations were observed for the depleted uranium capture reaction rates near the BeO plates. With time these issues should be addressed by adding the two- or three-dimensional modeling capabilities of MOCFE to MC²-3.

As a final note, an additional mesh merging function was added to UNIC to facilitate the unstructured mesh capability added to BuildZPRmodel and address the outstanding mesh generation issues with CUBIT. The MOCFE scaling studies made heavy use of this tool and we can expect that to continue unless the mesh related problems are addressed.

8 References

1. A. Siegel, T. Tautges, A. Caceres, D. Kaushik, and P. Fischer, "Software Design of SHARP," *Proc. Joint Int'l Topical Mtg. on Mathematics & Computation and Supercomputing in Nuclear Applications*, Monterey, California, April 15–19, 2007.
2. M. Smith, C. Lee, A. Wollaber, K. Derstine, D. Kaushik, E. Lewis, A. Mohamed, W. Yang. FY2009 Status Report on Advanced Neutronics Modeling and Validation, ANL-AFCI-282, Sept. 2009.
3. Micheal A. Smith, Dinesh Kaushik, Allan Wollaber, Won Sik Yang, Barry Smith, Cristian Rabiti, Giuseppe Palmiotti, "Recent Research Progress on UNIC at Argonne National Laboratory," International Conference on Mathematics, Computational Methods & Reactor Physics, Saratoga Springs, New York, May 3-7, 2009.
4. D.P. Weber et al, "High Fidelity LWR Analysis with the Numerical Nuclear Reactor," *Nuclear Science and Engineering*, **155**, 395-408 (2007).
5. C. H. Lee and W. S. Yang, "Status Report on Multi-group Cross Section Generation Code Development for High-Fidelity Deterministic Neutronics Simulation System," ANL-AFCI-207, Argonne National Laboratory, September 2007.
6. B. J. Toppel, H. Henryson II, and C. G. Stenberg, "ETOE-2/MC2-2/SDX Multi-group Cross-Section Processing," Conf-780334-5, Presentation at RSIC Seminar Workshop on Multi-group Cross Sections, Oak Ridge, TN, March 1978.
7. G. Palmiotti, E. E. Lewis & C. B. Carrico, "VARIANT: VARIational Anisotropic Nodal Transport for Multidimensional Cartesian and Hexagonal Geometry Calculation," Argonne National Laboratory ANL-95/40, 1995.

8. Derstine, K. L., "DIF3D: A Code to Solve One-, Two-, and Three-Dimensional Diffusion Theory Problems," ANL-82-64, Argonne National Laboratory. 1982.
9. S. Balay, K. R. Buschelman, W. D. Gropp, D. K. Kaushik, M. G. Knepley, L. C. McInnes, and B. F. Smith, "PETSc home page," www.mcs.anl.gov/petsc.
10. Personal Communication, Tom Manteuffel, Allan Wollaber, Micheal A Smith, et. al., University of Colorado at Boulder, October 2009.
11. Yair Shapira, "Algebraic multigrid," Matrix-based multigrid: theory and applications." Springer. ISBN 1402074859 (2003).
12. C. Rabiti, M. A. Smith, G. Palmiotti, "A Three-dimensional Method of Characteristics on Unstructured Tetrahedral Meshes," Trans. Am. Nucl. Soc. 96, 470, 2007.
13. Suslov I. R., "An Algebraic Collapsing Acceleration in Long Characteristics Transport Theory," Proc. 11th Symp. Atomic Energy Research, Csopak, Hungary, 2001.
14. Y. Saad and M.H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems", SIAM J. Sci. Stat. Comput., 7:856-869, 1986.
15. Personal Communication, Marvin Adams and Micheal A Smith, Pittsburgh PA, June, 2010.
16. Jonathan Richard Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," Applied Computational Geometry: Towards Geometric Engineering, volume 1148 of Lecture Notes in Computer Science, pages 203-222, Springer-Verlag, Berlin, May 1996.
17. Y. Saad, Iterative Methods for sparse Linear Systems, 2nd Ed. SIAM, Philadelphia, 2003
18. W. E. Arnoldi, "The Principle of Minimized Iteration in the Solution of the Matrix Eigenvalue Problem," Quart. Appl. Math. 9 17-29 (1951).
19. E. E. Lewis, A. Wollaber, A. Marin-Lafleche, M. A. Smith, and W. S. Yang, "Response Matrix Acceleration Methods Based on Orthogonalization and Domain Decomposition," Trans. Am. Nucl. Soc., 101. 360-361 (2010)
20. Personal Communication, Richard M. Lell and Micheal A Smith, Argonne National Laboratory, January 2009.
21. Oak Ridge National Laboratory, The Radiation Safety Informational Computer Center, "RSICC web page," <http://www-rsicc.ornl.gov/>.
22. Argonne National Laboratory, Argonne Leadership Computing Facility, <http://www.alcf.anl.gov>.
23. Oak Ridge National Laboratory, National Center for Computational Sciences, <http://www.nccs.gov>.
24. Jülich Research Centre, Jugene supercomputer, Germany, <http://www.fz-juelich.de/jsc/jugene>.
25. NEA Nuclear Science Committee, "International Handbook of Evaluated Criticality Safety Benchmark Experiments," NEA/NSC/DOC(95)03, September 2007.
26. D. C. Wade, "Monte Carlo based Validation of the ENDF/MC²-II/SDX Cell Homogenization Path," ANL-79-5, April 1979.
27. W. S. Yang and S. J. Kim, "A Validation Study of Existing Neutronics Tools Against ZPPR-21 and ZPPR-15 Critical Experiments," ANL-AFCI-208, September 2007.
28. Richard M. Lell, James A. Morman, Robert W. Schaefer and Richard D. McKnight, "ZPR-6 Assembly 7 High ²⁴⁰Pu Core Experiments: A Fast Reactor Core With Mixed (Pu,U)-Oxide Fuel

-
- and a Central High ^{240}Pu Zone.” ZPR-LMFR-EXP-002, CRIT-REAC-RRATE, NEA/NSC/DOC(1006)1.
29. VisIt. Website, Lawrence Livermore National Laboratory. <http://www.llnl.gov/visit>.
 30. Brian Warner. *BuildBot Manual 0.7.11*. Project web page at <http://buildbot.net/> [accessed September 21, 2009].
 31. W. R. Robinson and G. S. Stanford, “Effect of Simulated B_4C Control Rods on Reaction Rate Distribution in Assembly 7 with the High 240 Plutonium Zone,” ZPR-TM-74, Argonne National Laboratory, 1971.



Nuclear Engineering Division

Argonne National Laboratory
9700 South Case Avenue – Bldg. 208
Argonne, IL 60439

www.anl.gov