

# ABSTRACT

HOAGLAND, DYLAN SCOTT. A Family of Hybrid *Parallel Block Jacobi*-Type Iterative Methods & a Novel Green's Function Algorithm for Massively Parallel Solution of the Neutron Transport Equation on Unstructured Grids. (Under the direction of Dr. Yousry Y. Azmy).

The growing number of processors on modern-day supercomputers has introduced the challenge of developing methods and algorithms for effective scientific simulations on the class of massively parallel computer architectures. In the field of neutron transport specifically, spatially parallel solution on unstructured grids using the traditional *Source Iteration (SI)* method invokes scheduling difficulties, which *Parallel Block Jacobi (PBJ)*-type methods present an alternative to. This work presents two primary lines of work on *PBJ* methods with the objective of developing methods and algorithms that are applicable to the solution of the transport equation on unstructured grids using massively parallel supercomputers.

The first line of work pertains to the development, analysis, and testing of hybrid methods designed to improve the iterative performance of *PBJ* methods in problems containing optically thin cells. The iterative slowdown incurred by *PBJ* methods in such problems is a consequence of the resulting algorithm's asynchronicity. Hence, we propose a set of hybrid approaches which utilize a specified combination of *PBJ*-type and sweep-based methods to solve a given problem. Spectral analysis of the sequential execution of *Parallel Block Jacobi-Integral Transport Matrix Method (PBJ-ITMM)* or *Inexact Parallel Block Jacobi (IPBJ)* and *SI* demonstrates this approach to achieve iterative robustness in optically thin cells, but suggests that global execution of both methods is unnecessary. Ensuing numerical tests in serial operation on 2-D Cartesian grids, including a parametric study and testing on benchmark problems containing void regions, demonstrates that executing *PBJ-ITMM* in optically thick regions and *IPBJ* in optically thin regions greatly increases the resulting iterative scheme's rate of convergence, while not impacting its potential for efficient parallel solution on unstructured grids. Serial operation on 2-D Cartesian grids is used for these studies to expedite the development process.

The second line of work concerns the development of an algorithm for execution of *PBJ-ITMM* on unstructured grids. Solution on unstructured grids via *PBJ-ITMM* requires construction of the associated response matrices. In this work, we address this challenge by developing a novel

*Green's Function ITMM Construction (GFIC)* algorithm, which constructs the *ITMM* matrices using pre-existing transport code mesh sweeps with an artificial set of problem parameters designed to facilitate construction of the desired matrices. Using this algorithm, we implement *PBJ-ITMM* on 3-D unstructured tetrahedral grids in the THOR code for parallel computation using nonblocking MPI communication. The performance of *PBJ-ITMM* compared to *IPBJ* is then tested in this configuration with strong and weak scaling tests on up to 32,768 processors. The comparison to *IPBJ* is made, as *IPBJ* is the currently used alternative to spatially parallel *SI* on unstructured grids. These scaling tests are executed on two benchmark problems, Godiva and C5G7. The results of these scaling tests establish the applicability of *PBJ-ITMM* to the massively parallel solution of the transport equation on unstructured grids, solving a problem with >6.8 billion unknowns on 32,768 processors in under 20 minutes. The better performing of the two methods studied is observed to be dependent on the size of the sub-domains *PBJ* decomposes a spatial mesh into, that, in turn determines the rank of the applicable *ITMM* matrices for that sub-domain. *PBJ-ITMM* requires shorter solution time than *IPBJ* when sub-domains are comprised of fewer cells. The threshold at which this transition occurs is problem-dependent, occurring at ~128 and over 256 cells per sub-domain for Godiva and C5G7, respectively. The increased threshold for C5G7 compared to Godiva was due to C5G7 being a more computationally difficult problem, indicating that *PBJ-ITMM* is preferable over a larger range of sub-domain sizes in problems most in need of parallelization. These results lead to the conclusion that *PBJ-ITMM* is well suited for massively parallel solution on unstructured grids, and that *GFIC* is a viable construction algorithm for its implementation. At large processor counts, *GFIC*'s construction time was mostly insignificant compared to the total solution time, and the use of pre-existing mesh-sweep capabilities in the target transport code makes it a practical approach for current production transport codes utilizing the *SI* iterative solution method.

© Copyright 2020 by Dylan S. Hoagland

All Rights Reserved

A Family of Hybrid *Parallel Block Jacobi*-Type Iterative Methods & a Novel Green's Function  
Algorithm for Massively Parallel Solution of the Neutron Transport  
Equation on Unstructured Grids

by  
Dylan Scott Hoagland

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

Nuclear Engineering

Raleigh, North Carolina  
2020

APPROVED BY:

---

Dr. Yousry Azmy  
Committee Chair

---

Dr. Dmitriy Anistratov

---

Dr. Kostadin Ivanov

---

Dr. Carl Kelley

---

Dr. Robert Zerr

# **DEDICATION**

To my parents, my brother, and my fiancée.

# BIOGRAPHY

Dylan S. Hoagland was born on June 22, 1991 in Murfreesboro, TN. He and his brother Tristan were raised by their parents, Mary and Gordon, in Wilmington, NC. As a young boy, his interest in science was piqued watching PBS specials on String Theory and Quantum Mechanics, fascinated by the deductions into the very nature of reality made simply with a pen and paper. Upon entering into adulthood, Dylan pursued a degree in Nuclear Engineering, following a long and fruitless foray into nearly every other conceivable degree. Near the end of his undergraduate career, he was approached by then department head Dr. Azmy, who presented him with the offer of grad school. After months of self-deliberation, Dylan hesitantly accepted this offer. To date, it remains one of the best decisions he has ever made. In the pursuit of his PhD he found, not only a meaningful and exciting line of work, but a wonderful woman to spend his life with. Despite constantly envisioning over the years how his ten years at NC State would come to a close, he failed to ever predict that it would end in the den of his apartment, giving his defense remotely, under quarantine during the COVID-19 pandemic. Afterwards, in lieu of the long-expected celebrations, pomp and circumstance, he and his fiancée quietly left the state they had long known as home, leaving a note in Dylan's dissertation biography so that each person they were unable to say a proper goodbye to would know they were in their thoughts.

# ACKNOWLEDGMENTS

The work of the author is based upon work supported by the Department of Energy National Nuclear Security Administration under Award Number DE-NA0002576.

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 any of their employees, 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, endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This research made use Idaho National Laboratory computing resources which are supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517.

The author would like to personally acknowledge Drs. Yousry Azmy, Dmitriy Anistratov, Joe Zerr, and Sebastian Schunert for their incredible teaching and mentorship.

# TABLE OF CONTENTS

LIST OF TABLES .....	vii
LIST OF FIGURES .....	x
Chapter 1: Introduction .....	1
1.1: Discretization of the Neutron Transport Equation.....	3
1.2: The <i>Source Iteration</i> Method for Neutron Transport Calculations .....	6
1.3: <i>Parallel Block Jacobi</i> Spatial Domain Decomposition .....	7
1.4: Dissertation Outline .....	9
1.5: Terminology.....	11
Chapter 2: Literature Review .....	13
2.1: Acceleration of Neutron Transport Methods .....	13
2.1.1: <i>Diffusion Synthetic Acceleration (DSA)</i> .....	14
2.1.2: <i>Adjacent-Cell Preconditioning (AP)</i> .....	23
2.1.3: Coarse Mesh Rebalance.....	29
2.1.4: <i>Quasidiffusion (QD)</i> .....	31
2.1.5: <i>Nonlinear Diffusion Acceleration (NDA)</i> .....	33
2.1.6: <i>Transport Synthetic Acceleration (TSA)</i> .....	40
2.2: Parallel Transport Computations on Cartesian and Unstructured Grids.....	42
2.2.1: Angular Domain Decomposition .....	43
2.2.2: Energy Domain Decomposition.....	44
2.2.3: <i>Koch-Baker-Alcouffe (KBA)</i> Spatial Domain Decomposition on Cartesian Grids .....	45
2.2.4: Parallel Mesh Sweeping on Unstructured Grids.....	48
2.2.5: <i>Parallel Block Jacobi (PBJ)</i> Spatial Domain Decomposition .....	50
2.2.6: Iterative Methods for the <i>PBJ</i> Spatial Domain Decomposition .....	51
2.2.7: Iterative Convergence of Methods Using <i>PBJ</i> Spatial Domain Decomposition.....	54
2.2.8: Acceleration of <i>PBJ</i> -Type Methods .....	55
2.2.9: Hybrid Domain Decomposition.....	56
Chapter 3: <i>P-PI-SI &amp; P-IP-SI</i> Spectral Analysis & Formulation of the <i>Green's Function</i> <i>ITMM</i> Construction Algorithm.....	57
3.1 <i>P-PI-SI &amp; P-IP-SI</i> Theoretical Analysis.....	57
3.1.1 Formulation of <i>P-PI-SI &amp; P-IP-SI</i> .....	58
3.1.2 <i>P-PI-SI</i> Spectral Analysis .....	62
3.1.3 <i>P-IP-SI</i> Spectral Analysis .....	68
3.2 <i>ITMM</i> Matrices Construction Algorithms .....	73
3.2.1 <i>ITMM</i> Matrix Equation Formulation .....	74
3.2.2 <i>ITMM</i> Matrix Elements.....	77
3.2.3 <i>Differential Mesh Sweep (DMS)</i> .....	79
3.2.4 <i>Green's Function ITMM Construction (GFIC)</i> Algorithm .....	86
3.2.5 Computational Cost of <i>ITMM</i> Matrix Construction Algorithms .....	90

3.3 Summary of Theoretical Analysis .....	92
Chapter 4: Numerical Experimentation with Hybrid Methods in 2-D Cartesian Geometry with Serial Operation .....	94
4.1: HAT-2C Code Verification .....	97
4.2: <i>P-PI-SI</i> & <i>P-IP-SI</i> Fourier Analysis Results .....	103
4.3: Eigenvalues of <i>PBJ</i> methods as functions of fundamental transport phenomena .....	110
4.4: Comparison of <i>P-PI-SI</i> & <i>P-IP-SI</i> to their individual iterative constituents .....	112
4.5: <i>P-PI-SI</i> & <i>P-IP-SI</i> in Heterogeneous Problems and the Asynchronous Hybrid Approach .....	116
4.6: Parametric Study of Hybrid Methods .....	121
4.7: Parametric Study with Extreme Scattering Ratios.....	143
4.8: Test Problems Containing Void Regions.....	151
4.9: Computational Cost Comparison of <i>ITMM</i> Matrix Construction Algorithms .....	158
4.10: Summary of Serial Operation Numerical Experiments on 2-D Cartesian Grids.....	160
Chapter 5: <i>GFIC</i> Construction & <i>PBJ</i> Solution in Parallel Execution on 3-D Unstructured Tetrahedral Grids .....	162
5.1: THOR Mesh Decomposition Pre-Processing .....	163
5.2: <i>GFIC</i> Implementation in THOR .....	166
5.3: <i>PBJ-ITMM</i> and <i>IPBJ</i> Implementation in THOR .....	169
5.4: MPI Implementation for Processor Communication .....	173
5.5: <i>PBJ-ITMM</i> Memory Requirements .....	176
5.6: <i>PBJ-ITMM</i> and <i>IPBJ</i> Scaling Tests with THOR.....	179
5.6.1: Godiva Scaling Results.....	180
5.6.2: C5G7 Scaling Results .....	203
5.7: CPU Computation Time Comparison to Synchronously Sweeping.....	225
5.8: Iterative Performance on Problem Containing Void Region.....	229
5.9: Summary of Parallel Numerical Experiments on 3-D Unstructured Grids .....	234
Chapter 6: Conclusions & Future Work .....	237
6.1: Conclusions of Hybrid Combinations of <i>PBJ-Type</i> and Sweep-Based Methods.....	237
6.2: Conclusions of <i>GFIC</i> , <i>PBJ-ITMM</i> , & <i>IPBJ</i> in Parallel Execution on Unstructured Grids .....	240
6.3: Proposed Future Work.....	242
References.....	245
Appendices.....	254
Appendix A: Full Results from Parametric Study in Section 4.6.....	255
Appendix B: Full Results from Extreme Scattering Ratio Parametric Study in Section 4.7.....	296
Appendix C: Full Results from THOR Godiva Testing Suite in Table 5.2.....	336
Appendix D: Full Results from THOR C5G7 Testing Suite in Table 5.4.....	349

# LIST OF TABLES

Table 4.1: Iteration counts required by HAT-2C to converge the problem depicted in Fig. 4.1 for the homogeneous case using <i>SI</i> and <i>DSA</i> for multiple scattering ratios. $\Sigma_t \Delta u = 1.0$ , $u = x$ or $y$ , relative stopping criterion: $\epsilon = 10^{-6}$ .	98
Table 4.2: Iteration counts required by HAT-2C to converge the problem depicted in Fig. 4.1 for the homogeneous case using <i>SI</i> and <i>DSA</i> for multiple cell sizes. $c = 0.98$ , relative stopping criterion: $\epsilon = 10^{-6}$ .	98
Table 4.3: Iteration counts required by HAT-2C to converge the problem depicted in Fig. 4.1 for the heterogeneous case using <i>SI</i> and <i>DSA</i> and the associated cross sections for the three materials. $\Delta u = 2.0$ , $u = x$ or $y$ , relative stopping criterion: $\epsilon = 10^{-5}$ .	99
Table 4.4: Iteration counts required by HAT-2C to converge the problem depicted in Fig. 4.2 using <i>AP</i> for multiple values of $\Sigma_{t,1}$ , $\Sigma_{t,2}$ , $\Delta y_1$ , and $\Delta y_2$ . All other problem parameters are fixed as indicated in Fig. 4.2.	100
Table 4.5: Number of iterations required to converge the heterogeneous striped problem depicted in Fig. 4.23 with increasing $c$ , using the various iterative strategies and <i>AHOT-N0</i> . Relative iteration stopping criterion: $10^{-6}$ , $S_6$ angular quadrature. <i>PBJ</i> methods use one cell per sub-domain.	119
Table 4.6: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 16$ , $\Sigma_{thin} \Delta u_{thin} = 0.01$ , $\Sigma_{thick} \Delta u_{thick} = 5$ (Base Case)	144
Table 4.7: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 4$ , $\Sigma_{thin} \Delta u_{thin} = 0.01$ , $\Sigma_{thick} \Delta u_{thick} = 5$	144
Table 4.8: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 8$ , $\Sigma_{thin} \Delta u_{thin} = 0.01$ , $\Sigma_{thick} \Delta u_{thick} = 5$	145
Table 4.9: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 32$ , $\Sigma_{thin} \Delta u_{thin} = 0.01$ , $\Sigma_{thick} \Delta u_{thick} = 5$	145
Table 4.10: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 64$ , $\Sigma_{thin} \Delta u_{thin} = 0.01$ , $\Sigma_{thick} \Delta u_{thick} = 5$	146

Table 4.11: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 5$ .....	146
Table 4.12: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 5$ .....	147
Table 4.13: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 5$ .....	147
Table 4.14: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 1$ .....	148
Table 4.15: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 2.5$ .....	148
Table 4.16: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 10$ .....	149
Table 4.17: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios: $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 1$ .....	149
Table 4.18: Iterations required to converge Center Void Problem.....	152
Table 4.19: Runtime observed in seconds to converge Center Void Problem .....	153
Table 4.20: Iterations required to converge Straight Duct Problem.....	155
Table 4.21: Runtime observed in seconds to converge Straight Duct Problem.....	156
Table 4.22: Iterations required to converge Dog Leg Duct Problem.....	157
Table 4.23: Runtime observed in seconds to converge Dog Leg Duct Problem .....	157
Table 5.1: RAM (GB) per processor consumed by THOR to solve Godiva using <i>PBJ-ITMM</i> with $N$ cells per sub-domain with varying quadrature orders.....	177
Table 5.2: Godiva scaling test suite .....	181

Table 5.3: Target and actual mesh sizes and percent of physical volume modeled for Godiva scaling test suite .....	182
Table 5.4: C5G7 scaling test suite .....	204
Table 5.5: Target and actual mesh sizes for C5G7 scaling test suite.....	205
Table 5.6: Number of required iterations to converge ~262,144 cell Godiva mesh.....	225
Table 5.7: Serial CPU time (minutes) consumed to converge ~262,144 cell Godiva mesh .....	225
Table 5.8: Number of required iterations to converge ~262,144 cell C5G7 mesh.....	226
Table 5.9: Serial CPU time (hours) consumed to converge ~262,144 cell C5G7 mesh.....	226
Table 5.10: THOR grind times ( $\mu s$ ) for <i>PBJ-ITMM</i> and <i>IPBJ</i> (same as <i>SI</i> ) .....	229
Table 5.11: Iterations required by <i>PBJ-ITMM</i> and <i>IPBJ</i> to converge the dog-leg duct problem with THOR for varying problem parameters .....	233

# LIST OF FIGURES

Figure 1.1: <i>PBJ</i> Spatial Domain Decomposition. Bold lines indicate sub-domain boundaries. Thin lines indicate cell boundaries.....	8
Figure 2.1: Sub-domains for a 4-processor <i>KBA</i> decomposition of 2-D Cartesian mesh. Left image indicates the order of solution for an angle in the first quadrant, right, the processors to which sub-domains are assigned .....	45
Figure 4.1: Setup of verification problem with varying $c$ and cell size for <i>SI</i> and <i>DSA</i> from [13].....	97
Figure 4.2: Setup of verification problem for <i>AP</i> from [20]. $\Delta x = 1.0$ .....	99
Figure 4.3: Computationally estimated spectral radius produced by HAT-2C for <i>PBJ-ITMM</i> versus cell size for multiple scattering ratios of a problem with an $N \times N$ mesh.....	101
Figure 4.4: Computationally estimated spectral radius produced by HAT-2C for <i>IPBJ</i> versus cell size for multiple scattering ratios of a problem with an $N \times N$ mesh. ....	102
Figure 4.5: <i>P-PI-SI</i> (left) and <i>P-IP-SI</i> (right) eigenvalues: $\Sigma_t = 0.1, c = 0.1, \Delta x = 1, \Delta y = 1$ .....	103
Figure 4.6: <i>P-PI-SI</i> (left) and <i>P-IP-SI</i> (right) eigenvalues: $\Sigma_t = 0.1, c = 0.5, \Delta x = 1, \Delta y = 1$ .....	104
Figure 4.7: <i>P-PI-SI</i> (left) and <i>P-IP-SI</i> (right) eigenvalues: $\Sigma_t = 0.1, c = 0.9, \Delta x = 1, \Delta y = 1$ .....	104
Figure 4.8: <i>P-PI-SI</i> (left) and <i>P-IP-SI</i> (right) eigenvalues: $\Sigma_t = 1, c = 0.1, \Delta x = 1, \Delta y = 1$ .....	104
Figure 4.9: <i>P-PI-SI</i> (left) and <i>P-IP-SI</i> (right) eigenvalues: $\Sigma_t = 1, c = 0.5, \Delta x = 1, \Delta y = 1$ .....	105
Figure 4.10: <i>P-PI-SI</i> (left) and <i>P-IP-SI</i> (right) eigenvalues: $\Sigma_t = 1, c = 0.9, \Delta x = 1, \Delta y = 1$ .....	105
Figure 4.11: <i>P-PI-SI</i> (left) and <i>P-IP-SI</i> (right) eigenvalues: $\Sigma_t = 10, c = 0.1, \Delta x = 1, \Delta y = 1$ .....	105

Figure 4.12: $P$ - $PI$ - $SI$ (left) and $P$ - $IP$ - $SI$ (right) eigenvalues: $\Sigma_t = 10, c = 0.5, \Delta x = 1, \Delta y = 1$ .....	106
Figure 4.13: $P$ - $PI$ - $SI$ (left) and $P$ - $IP$ - $SI$ (right) eigenvalues: $\Sigma_t = 10, c = 0.9, \Delta x = 1, \Delta y = 1$ .....	106
Figure 4.14: $P$ - $PI$ - $SI$ (left) and $P$ - $IP$ - $SI$ (right) eigenvalues: $\Sigma_t = 1, c = 0.5, \Delta x = 1, \Delta y = 0.1$ .....	106
Figure 4.15: $P$ - $PI$ - $SI$ (left) and $P$ - $IP$ - $SI$ (right) eigenvalues: $\Sigma_t = 1, c = 0.5, \Delta x = 10, \Delta y = 0.1$ .....	107
Figure 4.16: Spectral radius of $P$ - $PI$ - $SI$ estimated with HAT-2C for a homogeneous problem with an $N \times N$ mesh and the theoretically calculated spectral radius. Spectral radius is graphed versus cell size for multiple scattering ratios using an $S_6$ angular quadrature. ....	108
Figure 4.17: Spectral radius of $P$ - $IP$ - $SI$ estimated with HAT-2C for a homogeneous problem with an $N \times N$ mesh and the theoretically calculated spectral radius. Spectral radius is graphed versus cell size for multiple scattering ratios using an $S_6$ angular quadrature. ....	108
Figure 4.18: Spectral radius of $PBJ$ - $ITMM$ vs. $\Sigma_a$ and $\Sigma_s$ with $S_6$ angular quadrature .....	111
Figure 4.19: Spectral radius of $IPBJ$ vs. $\Sigma_a$ and $\Sigma_s$ with $S_6$ angular quadrature.....	114
Figure 4.20: Iterations required to converge $100 \times 100$ homogeneous problem: $c = 0.1$ .....	114
Figure 4.21: Iterations required to converge $100 \times 100$ homogeneous problem: $c = 0.5$ .....	115
Figure 4.22: Iterations required to converge $100 \times 100$ homogeneous problem: $c = 0.9$ .....	115
Figure 4.23: Layout of heterogeneous stripe problem with uniform unit source. Thick regions contain $10$ $mfp$ cells, thin regions contain $0.1$ $mfp$ cells.....	117
Figure 4.24: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$ (Base Case).....	124
Figure 4.25: Runtime observed for parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$ (Base Case).....	124
Figure 4.26: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 4, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$ .....	125

Figure 4.27: Runtime observed for parametric study problem versus scattering ratio: $N_s = 4, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$ .....	125
Figure 4.28: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 8, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$ .....	126
Figure 4.29: Runtime observed for parametric study problem versus scattering ratio: $N_s = 8, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$ .....	126
Figure 4.30: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 32, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$ .....	127
Figure 4.31: Runtime observed for parametric study problem versus scattering ratio: $N_s = 32, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$ .....	127
Figure 4.32: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 64, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$ .....	128
Figure 4.33: Runtime observed for parametric study problem versus scattering ratio: $N_s = 64, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$ .....	128
Figure 4.34: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.001, \Sigma_{thick} \Delta u_{thick} = 5$ .....	129
Figure 4.35: Runtime observed for parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.001, \Sigma_{thick} \Delta u_{thick} = 5$ .....	129
Figure 4.36: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.1, \Sigma_{thick} \Delta u_{thick} = 5$ .....	130
Figure 4.37: Runtime observed for parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.1, \Sigma_{thick} \Delta u_{thick} = 5$ .....	130
Figure 4.38: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.5, \Sigma_{thick} \Delta u_{thick} = 5$ .....	131
Figure 4.39: Runtime observed for parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.5, \Sigma_{thick} \Delta u_{thick} = 5$ .....	131

Figure 4.40: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 1$ .....	132
Figure 4.41: Runtime observed for parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 1$ .....	132
Figure 4.42: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 2.5$ .....	133
Figure 4.43: Runtime observed for parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 2.5$ .....	133
Figure 4.44: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 10$ .....	134
Figure 4.45: Runtime observed for parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 10$ .....	134
Figure 4.46: Iterations required to converge parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.1, \Sigma_{thick} \Delta u_{thick} = 1$ .....	135
Figure 4.47: Runtime observed for parametric study problem versus scattering ratio: $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.1, \Sigma_{thick} \Delta u_{thick} = 1$ .....	135
Figure 4.48: Geometry, mesh, and problem parameters for the Center Void Problem .....	152
Figure 4.49: Geometry, mesh, and problem parameters for the Straight Duct Problem .....	155
Figure 4.50: Geometry, mesh, and problem parameters for the Dog Leg Duct Problem .....	156
Figure 4.51: Measured (points) and predicted (lines) construction times for <i>ITMM</i> matrices using <i>HAT-2C</i> for a single sub-domain using <i>DMS</i> and <i>GFIC</i> (ideal and full sweep) versus $N$ , indicating an $N \times N$ sub-domain size. ....	159
Figure 5.1: Flow chart of process to create partitioned THOR mesh files .....	164
Figure 5.2: Flow chart of THOR mesh partitioner tool .....	165
Figure 5.3: Flow chart of <i>PBJ-ITMM</i> inner iteration process on a single processor in THOR .....	170

Figure 5.4: Flow chart of <i>IPBJ</i> inner iteration per process in THOR.....	172
Figure 5.5: Godiva tetrahedral mesh with ~32 cells .....	183
Figure 5.6: Godiva tetrahedral mesh with ~512 cells .....	184
Figure 5.7: Godiva tetrahedral mesh with ~16,384 cells .....	184
Figure 5.8: Godiva ~16,384 cell mesh, partitioned over 16 sub-domains .....	185
Figure 5.9: Godiva ~16,384 cell mesh, partitioned over 512 sub-domains .....	186
Figure 5.10: Godiva ~16,384 cell mesh, partitioned over 2048 sub-domains .....	186
Figure 5.11: <i>k-eigenvalues</i> for Godiva produced by parallel THOR vs. mesh size.....	187
Figure 5.12: Time consumed by <i>GFIC</i> to construct <i>ITMM</i> matrices for Godiva in THOR (points) and leading order of theoretical construction time, Eq. (3.88) (line) .....	188
Figure 5.13: Iteration strong scaling for Godiva: ~32,768 cell mesh .....	190
Figure 5.14: Solution time strong scaling for Godiva: ~32,768 cell mesh .....	190
Figure 5.15: Iteration strong scaling for Godiva: ~65,536 cell mesh .....	191
Figure 5.16: Solution time strong scaling for Godiva: ~65,536 cell mesh .....	191
Figure 5.17: Iteration strong scaling for Godiva: ~131,072 cell mesh .....	192
Figure 5.18: Solution time strong scaling for Godiva: ~131,072 cell mesh .....	192
Figure 5.19: Iteration strong scaling for Godiva: ~262,144 cell mesh .....	193
Figure 5.20: Solution time strong scaling for Godiva: ~262,144 cell mesh .....	193
Figure 5.21: Iteration weak scaling for Godiva: ~32 cells per sub-domain (processor).....	198
Figure 5.22: Solution time weak scaling for Godiva: ~32 cells per sub-domain (processor).....	198
Figure 5.23: Iteration weak scaling for Godiva: ~64 cells per sub-domain (processor).....	199
Figure 5.24: Solution time weak scaling for Godiva: ~64 cells per sub-domain (processor).....	199

Figure 5.25: Iteration weak scaling for Godiva: ~128 cells per sub-domain (processor).....	200
Figure 5.26: Solution time weak scaling for Godiva: ~128 cells per sub-domain (processor).....	200
Figure 5.27: Iteration weak scaling for Godiva: ~256 cells per sub-domain (processor).....	201
Figure 5.28: Solution time weak scaling for Godiva: ~256 cells per sub-domain (processor).....	201
Figure 5.29: C5G7 Benchmark problem’s mesh with ~524,288 tetrahedral cells.....	206
Figure 5.30: C5G7 Benchmark problem’s mesh with ~1,048,576 tetrahedral cells.....	206
Figure 5.31: C5G7 Benchmark problem’s mesh with ~2,097,152 tetrahedral cells.....	207
Figure 5.32: C5G7 mesh material legend .....	207
Figure 5.33: C5G7 Benchmark problem’s tetrahedral mesh at fuel assembly, moderator interface .....	208
Figure 5.34: Tetrahedral mesh of a C5G7 Benchmark problem’s fuel cell.....	209
Figure 5.35: C5G7 partitioned mesh, ~524,288 cells, 512 sub-domains.....	209
Figure 5.36: C5G7 partitioned mesh, ~524,288 cells, 1024 sub-domains.....	210
Figure 5.37: C5G7 partitioned mesh, ~524,288 cells, 2048 sub-domains.....	210
Figure 5.38: <i>k-eigenvalues</i> for C5G7 produced by THOR vs. mesh size.....	211
Figure 5.39: Time consumed by <i>GFIC</i> to construct <i>ITMM</i> matrices for C5G7 in THOR (points) and leading order of theoretical construction time, Eq. (3.88) (line) .....	212
Figure 5.40: Iteration strong scaling for C5G7: ~262,144 cell mesh .....	214
Figure 5.41: Solution time strong scaling for C5G7: ~262,144 cell mesh .....	214
Figure 5.42: Iteration strong scaling for C5G7: ~524,288 cell mesh .....	215
Figure 5.43: Solution time strong scaling for C5G7: ~524,288 cell mesh .....	215
Figure 5.44: Iteration strong scaling for C5G7: ~1,048,576 cell mesh .....	216
Figure 5.45: Solution time strong scaling for C5G7: ~1,048,576 cell mesh .....	216

Figure 5.46: Iteration strong scaling for C5G7: ~2,097,152 cell mesh .....	217
Figure 5.47: Solution time strong scaling for C5G7: ~2,097,152 cell mesh .....	217
Figure 5.48: Iteration weak scaling for C5G7: ~64 cells per sub-domain (processor).....	220
Figure 5.49: Solution time weak scaling for C5G7: ~64 cells per sub-domain (processor).....	220
Figure 5.50: Iteration weak scaling for C5G7: ~128 cells per sub-domain (processor).....	221
Figure 5.51: Solution time weak scaling for C5G7: ~128 cells per sub-domain (processor).....	221
Figure 5.52: Iteration weak scaling for C5G7: ~256 cells per sub-domain (processor).....	222
Figure 5.53: Solution time weak scaling for C5G7: ~256 cells per sub-domain (processor).....	222
Figure 5.54: <i>KBA</i> (left) and <i>PBJ</i> (right) decompositions of a 2-D Cartesian spatial domain for solution on four processors with communicating (solid) and non-communicating (dashed) sub-domain boundaries .....	227
Figure 5.55: Dog-leg void problem mesh .....	230
Figure 5.56: Dog-leg void problem partitioned mesh.....	230
Figure 5.57: Dog-leg void problem interior mesh .....	231
Figure 5.58: Dog-leg void problem interior partitioning .....	231
Figure 5.59: Dog-leg void problem duct region mesh.....	232
Figure 5.60: Dog-leg void problem duct region partitioning.....	232

# Chapter 1: Introduction

The neutron transport equation,

$$\begin{aligned}
 & \frac{1}{v(E)} \frac{\partial \psi(\vec{r}, \hat{\Omega}, E, t)}{\partial t} + \hat{\Omega} \cdot \vec{\nabla} \psi(\vec{r}, \hat{\Omega}, E, t) + \Sigma_t(\vec{r}, E) \psi(\vec{r}, \hat{\Omega}, E, t) & (1.1) \\
 & = \int_{4\pi} d\hat{\Omega}' \int_0^\infty dE' \Sigma_s(\vec{r}, E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(\vec{r}, \hat{\Omega}', E', t) \\
 & + \frac{\chi(E)}{4\pi} \int_0^\infty dE' \nu_f(\vec{r}, E') \Sigma_f(\vec{r}, E') \phi(\vec{r}, E', t) + q(\vec{r}, \hat{\Omega}, E, t), t > t_0, \vec{r} \\
 & \in D, 0 \leq E \leq \infty, \hat{\Omega} \in \text{Unit Sphere}
 \end{aligned}$$

is the mathematical model that describes the collective migration in phase space of neutrons in a specified physical configuration. [1] In this equation,  $v$  is the neutron speed,  $\Sigma_t$ ,  $\Sigma_s$ , and  $\Sigma_f$  are the total, scattering, and fission macroscopic cross sections (probability of interaction per unit distance traveled) respectively,  $\chi(E)$  is the energy distribution of fission neutrons,  $\nu_f$  is the average number of neutrons born in a fission event,  $D$  is the problem's spatial domain, and  $q$  is the external source of neutrons. The solution to this equation is  $\psi(\vec{r}, \hat{\Omega}, E, t)$ , such that  $\psi(\vec{r}, \hat{\Omega}, E, t) d\hat{\Omega} dE$  is the expected value of the angular flux of neutrons at location  $\vec{r}$ , traveling within  $d\hat{\Omega}$  about direction  $\hat{\Omega}$ , with energies  $dE$  about  $E$ , at time  $t$ . Another important quantity to note is  $\phi(\vec{r}, E, t)$ , the scalar flux, which is the angular flux integrated over all angles.

To solve the transport equation, Eq. (1.1) must be supplemented by boundary conditions and an initial condition. For the initial condition at time  $t_0$ , there must be a flux distribution,  $\psi(\vec{r}, \hat{\Omega}, E, t_0)$  that is known for all values of the other independent variables. This flux profile can be produced from multiple specifications for the beginning state of a problem, such as the steady-state solution of the problem before a perturbation or a restart file. Specifying boundary conditions is slightly more involved. In Eq. (1.1), the spatial gradient (second term on the left-hand-side) comprises a first order spatial derivative. This implies that a mathematical solution of the equation in Cartesian geometry only requires a single boundary condition for each spatial dimension of a

problem, defined for all values of the other independent variables. In practice, however, this is not generally known, as the outgoing angular flux at the problem boundary is dependent on the solution of the transport problem itself. The outgoing angular flux is the angular flux along directions  $\hat{\Omega}$  such that  $\hat{\Omega} \cdot \hat{n} > 0$ ,  $\hat{n}$  being the outward normal vector to the bounding surface. Therefore, the boundary conditions are split into “partial” boundary conditions where, in Cartesian geometry for each spatial dimension, there are two boundary conditions defined on opposite faces bounding the problem domain, each defining the angular flux only for directions where  $\hat{\Omega} \cdot \hat{n} < 0$  (incoming to the problem domain). This allows the neutron transport equation to be solved without requiring knowledge of the flux exiting the spatial problem domain, a quantity which is dependent on the transport solution.

In general, there are two main types of boundary conditions for the neutron transport equation, explicit and implicit. The former is the simpler of the two types, with the angular flux distribution explicitly provided at a boundary point for all  $\hat{\Omega}$  such that  $\hat{\Omega} \cdot \hat{n} < 0$ . A common special case of this type is the vacuum boundary condition, where the incoming angular flux distribution on the boundary is set to zero for  $\hat{\Omega} \cdot \hat{n} < 0$ , implying a scenario in which there are no impinging neutrons onto the problem domain, and where no neutrons which exit the domain subsequently re-enter it. [1, 2]

Contrarily, implicit boundary conditions substitute this explicit designation of the boundary incoming angular flux distribution with a mathematical relation to the boundary outgoing angular flux distribution. The most common types of implicit boundary conditions used are reflective (and more generally albedo) and periodic. A reflective boundary condition equates the incoming angular flux at a point on the boundary to the outgoing angular flux at that same spatial point, at the same energy, and with the associated reflected direction of the neutron’s flight path. Physically, this describes a scenario in which the problem configuration is reflectively symmetric about the corresponding boundary plane. This boundary condition is useful for modeling problems with reflective symmetry as it yields a smaller problem domain. [1, 2]

Additionally, the reflective boundary condition can be modified to an albedo boundary condition in which the incoming angular flux at a point on the boundary is set to a prespecified fraction of the outgoing angular flux at the same point in the reflected direction. This has been used to efficiently model a sub-region of a problem which impacts the problem’s bulk solution, but over which the solution is not of interest. Specifically, albedo boundary conditions can

effectively be used to model the reflector around a nuclear reactor core, simulating a certain percentage of outgoing neutrons reflecting back into the reactor core. This allows the solution process to focus computational resources on the important core region while accounting for the effect of the reflector region without having to solve the transport equation in the latter region. [3]

The other type of implicit boundary condition frequently used is the periodic boundary condition. This boundary condition equates the incoming angular flux distribution at a point on a boundary to the outgoing angular flux distribution at the corresponding point on the opposite, parallel boundary. This models a problem in which the spatial domain is a single instance of a larger, periodically repeated structure, e.g. fuel assemblies. [1, 2]

The solution to the neutron transport equation is needed for numerous applications in nuclear nonproliferation as well as other applications in nuclear engineering. It is often very difficult to obtain though, as Eq. (1.1) is an integro-differential equation, the solution to which has seven independent variables. Given the importance of transport solutions as well as the difficulty of computing them, development of methods and algorithms for obtaining accurate solutions to the transport equation in an efficient manner is an active area of research. While this work pertains to neutron transport, the concepts discussed extend more broadly to neutral particle transport, including gamma and x-ray transport.

## **1.1: Discretization of the Neutron Transport Equation**

Deterministically solving the transport equation requires the continuous form of Eq. (1.1) and its accompanying initial and boundary conditions to be transformed into a discrete linear system of equations by discretizing each of the independent variables. We will focus on discretization of the steady state transport equation (i.e. vanishing time derivative). The energy domain is traditionally discretized using the multigroup approach that amounts to discretizing the continuous energy domain into disjoint energy intervals, or groups. The group scalar and angular fluxes are defined as the corresponding energy dependent fluxes integrated over the span of the energy group. The group cross sections are then obtained by averaging the energy dependent cross sections over the range of the group using a suitable weighting spectrum. This invokes the inherent difficulty that obtaining adequate multigroup cross sections requires knowledge of the energy dependent angular flux, that is the solution of the transport equation that is sought. This circular

dependence is broken by fixing the weighting spectrum required to generate the multigroup constants to an energy distribution that is computed on a simplified configuration, e.g. reduced dimensionality. [1, 2]

There are two main approaches for discretization of the angular domain, the  $S_N$  and the  $P_N$  methods. In this work we will employ only the  $S_N$  method, which discretizes the angular domain into individual directions (discrete ordinates), requiring the neutron transport equation only to be satisfied for the selected directions. To obtain angularly integrated quantities, each discrete ordinate is assigned a quadrature weight. Any quantity to be integrated numerically over the angular domain (such as integrating the angular flux over  $\widehat{\Omega}$  to obtain the scalar flux) is then calculated as a weighted sum over all discrete ordinates. [2]

There exist numerous discretization methods for the spatial domain, [2] most of which can be classified into finite difference methods and finite element methods. In this work, we use a nodal method (that is akin to finite differencing methods) known as the *Arbitrary High Order Transport – Zeroth Order Nodal (AHOT-N0)* method, which is cast in a special form of the *Weighted Diamond Difference (WDD)* method.

To illustrate the spatial discretization formalism we consider, without loss of generality, a 1-D, 1-group, steady state transport problem with no fission, isotropic scattering, and isotropic external source. The discretized form of the neutron transport balance equation (1.1) is,

$$\mu_m \frac{\psi_{m,i}^+ - \psi_{m,i}^-}{\Delta x_i} + \Sigma_{t,i} \psi_{m,i} = \frac{1}{2} \Sigma_{s,i} \phi_i + \frac{1}{2} q_i \quad (1.2)$$

where  $\psi^+$  and  $\psi^-$  are the outgoing and incoming angular fluxes, respectively, to a spatial cell,  $\mu_m$  is the directional cosine with respect to the  $x$ -axis of the discrete ordinate  $m$ , and  $\Delta x_i$  is the size of the  $i^{\text{th}}$  spatial cell along the  $x$ -axis. Any flux quantity without a superscript is then a cell-averaged quantity. Assuming the incoming angular flux to be inferred from either the outgoing angular flux of an upstream neighbor or from the boundary condition, there are two distinct unknowns in this equation, requiring an auxiliary equation relating the incoming, outgoing, and average angular fluxes to solve simultaneously with Eq. (1.2).

In general, *WDD* methods employ an auxiliary equation of the form,

$$\psi_{m,i} = \frac{1 + \alpha_{m,i}}{2} \psi_{m,i}^+ + \frac{1 - \alpha_{m,i}}{2} \psi_{m,i}^- \quad (1.3)$$

where  $\alpha_{m,i}$  is the spatial weight for the *WDD* method, which is between zero and one. The distinctions between the various incarnations of *WDD* methods lie in the auxiliary equations needed for closure of the discretized linear algebraic system, more specifically in the choice of the spatial weights. In general, problems with cells of large optical thickness will tend to have larger weights, as the exponential attenuation through a cell makes the average more heavily weighted towards the outgoing face. Also, realistic problems will likely have cells of varying optical thicknesses, making a global weight less suitable. [2]

Methods such as *Diamond Difference (DD)* and *Step* use global weights of 0 and 1, respectively, while nodal methods calculate on the fly weights for each discrete ordinate, spatial cell combination. In this work, our focus is on iterative methods, so the selection of spatial discretization method is secondary. In our test code subsequently discussed, we implement the *AHOT-N0* method due to its ability to obtain a desired solution accuracy using fewer spatial cells than either of the aforementioned methods due to dynamic weights allowing for a coarser spatial mesh to be used. Additionally, the iterative methods we are interested in have previously been studied with *AHOT-N0*, providing results against which to verify. [4] The *AHOT-N0* method calculates the spatial weights using the expression, [5]

$$\alpha_{m,i} = \coth\left(\frac{\Sigma_{t,i}\Delta x_i}{2|\mu_m|}\right) - \frac{2|\mu_m|}{\Sigma_{t,i}\Delta x_i}. \quad (1.4)$$

In multidimensional configurations the *WDD* auxiliary equation, Eq. (1.3), is repeated once per dimension, and the corresponding spatial weight is computed independently in analogy to Eq. (1.4) using the corresponding cell size and angular cosine.

## 1.2: The *Source Iteration* Method for Neutron Transport Calculations

The discretization methods discussed in the previous section yield a closed linear system of equations for the entire problem. While direct solution of this system is theoretically possible by factorization or inversion of the effective matrix, doing so would be infeasibly time consuming and memory intensive for any problem of realistic size, as the rank of the applicable matrix operator becomes too large. Traditional iterative solution methods of transport problems use an inner / outer iterative strategy which uses the following general form of the discretized transport equation for a 1-D steady state problem,

$$\mu_m \frac{\psi_{g,m,i}^+ - \psi_{g,m,i}^-}{\Delta x_i} + \Sigma_{g,t,i} \psi_{g,m,i} = q_{g,i} + \sum_{g' \neq g} q_{g' \rightarrow g,i}. \quad (1.5)$$

In this general equation the subscript  $g$  and  $i$  indicate the energy group and spatial cell, respectively. On the right-hand-side,  $q_{g,i}$  contains the external source and the source of particles which scatter from the current group ( $g$ ) and remain in that energy group. This term is often referred to as the “within-group” scattering source.  $q_{g' \rightarrow g,i}$  is the source of particles which arrive in group  $g$  due to interaction by a parent neutron with energy in group  $g'$ .

The latter of these terms is the term to which the outer iterations apply. In an outer iteration, Eq. (1.5) is solved for all energy groups, one group at a time assuming all  $q_{g' \rightarrow g,i}$  terms are known. Depending on the details of the outer iteration scheme, parts of the  $q_{g' \rightarrow g,i}$  terms are inferred from the previous outer iteration’s flux, and the remaining parts are computed from the current outer iteration’s flux for all groups ( $g'$ ) which have already been solved.

With the  $q_{g' \rightarrow g,i}$  terms assumed known, and recognizing the dependence of  $q_{g,i}$  on  $\phi_{g,i}$ , this equation is still most practical to solve iteratively, resulting in the inner iterations. Our work addresses exclusively the inner iterations, and outer iterations will not even be presented until Chapter 5. Therefore, for brevity, in the entirety of this work the term “iterations” refers to the inner iterations, unless otherwise specified. Historically, these iterations work by lagging the within-group scattering term included in  $q_{g,i}$  by an iteration (i.e. setting the group flux used in

computing the within-group scattering term to the previous inner iteration's group flux). This method is termed the *Source Iteration (SI)* method. One *SI* iteration typically uses the mesh sweep algorithm for its solution. If we consider all terms on the right-hand-side of Eq. (1.5) to be known, then there are three unknown quantities in this equation for a given cell / discrete ordinate combination in one group,  $\psi_{g,m,i}^+$ ,  $\psi_{g,m,i}^-$ , and  $\psi_{g,m,i}$ . If we include the auxiliary equation, there are now three unknowns, but only two equations. However, for a boundary cell to which the angular flux of the current discrete ordinate is impinging, the incoming angular flux is known, either explicitly or implicitly, allowing for the outgoing and cell-averaged angular fluxes to be solved for directly. The downstream adjacent cell can then be solved using its upstream neighbor's outgoing angular flux as its incoming angular flux by applying flux continuity across that face. The entire spatial domain can then be "swept" in this manner for a given discrete ordinate. Performing a mesh sweep for each discrete ordinate starting from a boundary where the incoming flux is known, explicitly or iterating on the incoming angular flux on the boundary otherwise, constitutes one *SI* iteration. Note that while we have provided example equations for a 1-D problem, these same concepts are straightforward to extend to 2-D and 3-D problems.

With the within-group source assumed known, all angles in the problem are decoupled from each other within a single *SI*. Mathematically, the resulting systems of equations (one for each angle) comprise triangular matrices that can be solved by recursive substitution. The mesh sweep algorithm effectively performs this operation, yielding the solution of these matrix equations without constructing the subject matrices. [6]

### ***1.3: Parallel Block Jacobi Spatial Domain Decomposition***

While decoupling the angular domain for *SI* solution provides a simpler method for solving the within-group problem, in parallel the *Parallel Block Jacobi (PBJ)* spatial domain decomposition has also been used, decoupling portions of the spatial domain. A 2-D representation of the *PBJ* decomposition is depicted in Fig. 1.1. We see that the spatial domain is divided into multiple sub-domains, each of which contains multiple cells (although they may contain as few as 1 cell). From the zoomed-in sub-domain in the bottom left corner of the domain, we see that all incoming angular fluxes at the sub-domain boundary are considered known, and all outgoing angular fluxes at this boundary are unknown. The assumption that these incoming angular fluxes

are simultaneously known for all sub-domains is the defining factor of the *PBJ* decomposition. For these terms to be assumed as known, they must either be inferred from the boundary conditions or inferred from the outgoing angular fluxes of an adjacent sub-domain from the previous iteration. This implies that an iterative method must be used to obtain transport solutions using the *PBJ* decomposition until convergence of the sub-domain interface angular fluxes is achieved, although frequently control of the iterative sequence is based exclusively on testing convergence of the cell-averaged scalar fluxes. For the *PBJ* decomposition, the sub-domain interface angular fluxes are the only quantities which must be lagged, however, certain iterative methods lag other terms, e.g. the scattering source. For the *Parallel Block Jacobi-Integral Transport Matrix Method (PBJ-ITMM)* method, these incoming angular fluxes on sub-domain interfaces are the only lagged quantities. *Inexact Parallel Block Jacobi (IPBJ)* on the other hand, additionally lags the scattering source.

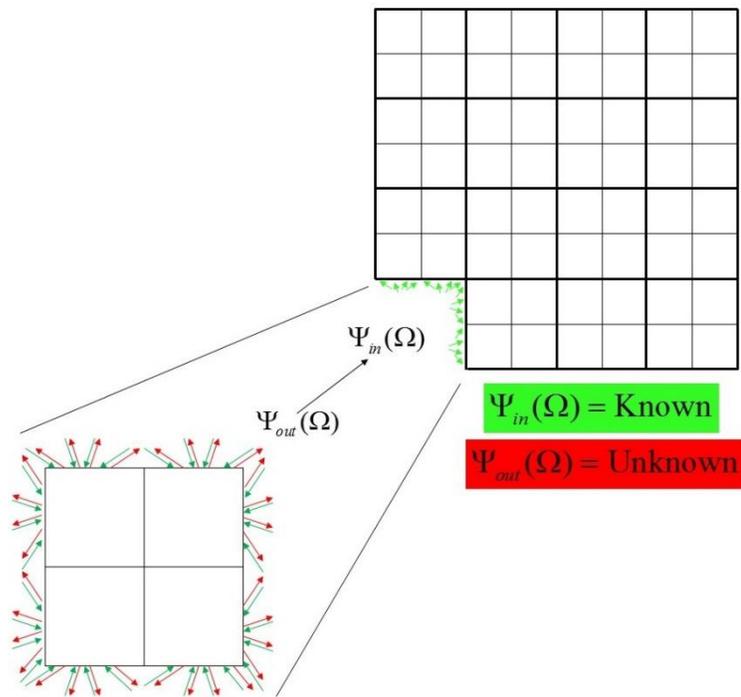


Figure 1.1: *PBJ* Spatial Domain Decomposition. Bold lines indicate sub-domain boundaries. Thin lines indicate cell boundaries.

## 1.4: Dissertation Outline

The motivation for the work presented in this dissertation is to develop, analyze, implement, and test methods and algorithms that are effective for deterministically solving the neutron transport equation on unstructured grids (meshes) on massively parallel supercomputers using *Parallel Block Jacobi (PBJ)*-type methods. In this endeavor, there are two main lines of development for this work: 1) hybrid methods that pair *PBJ*-type methods with mesh sweep-based methods to improve the iterative performance of *PBJ* in problems containing optically thin cells, while maintaining scalability to massively parallel supercomputers on unstructured grids and 2) an algorithm for construction of the matrices required for solution via *PBJ-ITMM* that is applicable to unstructured grids. Analysis and testing of the former is conducted in serial operation on Cartesian, 2-D grids with the assumption that the observed trends extend to parallel solution on unstructured grids. This simplified testing configuration is chosen for the development of our hybrid methods, as it expedites the implementation and data collection processes, resulting in an efficient development process. Testing for the latter of our two lines of work is conducted in parallel execution on unstructured grids. The complexity of this configuration is necessary for this portion of our work, as its purpose is to provide algorithms for the implementation on unstructured grids of all constituent methods that are used in our hybrid approaches. We outline this dissertation's contents as follows.

Already in the Introduction, we described the general process by which the neutron transport equation is discretized from its continuous form to obtain a linear system of algebraic equations. We also discussed the *SI* iterative method, which together with the mesh sweep algorithm is the standard method used for solving the per-group linear system of equations without construction and factorization of the group's full matrix equation, which is infeasibly large for almost any realistic problem. *PBJ*-type methods have also been introduced as alternatives to the *SI* iterative method.

In Chapter 2 we will proceed with a review of the relevant literature. We first discuss the mathematical analysis which is used to assess the convergence rate of iterative methods and the known properties of the *SI* method. We will then discuss several acceleration methods previously developed and currently used to improve *SI*'s rate of convergence. Finally, we will discuss parallel transport calculations on both Cartesian and unstructured spatial meshes, and the motivation for

implementing *PBJ*-type methods for parallel solution on the latter, as such applications constitute our research's ultimate objective.

Upon concluding the literature review, we separate the bulk of this document into three segments, beginning with theoretical analysis of our new hybrid methods and mathematical formulation of our novel algorithm for construction of the *ITMM* matrices on unstructured grids, presented in Chapter 3. Then in Chapter 4 we describe the numerical experiments we performed to assess our hybrid approach's performance in serial execution on 2-D Cartesian grid configurations. Chapter 5 is dedicated to reporting the parallel execution of *PBJ-ITMM* on unstructured grids using our novel algorithm for construction of the required matrices.

Theoretical analysis (Chapter 3) describes the Fourier analysis conducted on both *PBJ-ITMM* and *IPBJ* preconditioned with *SI*, which are the most basic of our hybrid approaches, designed for analysis to verify the efficacy of the fundamental approach. This analysis is followed by formulation of the matrix equations that are solved each *PBJ-ITMM* iteration, with our developed algorithm detailed for the matrices' construction.

Results of numerical testing of our hybrid approach then proceed in Chapter 4, beginning with verification of our 2-D, structured grids, serial operation transport code, HAT-2C. We then verify our Fourier analysis and conduct a set of tests on homogenous problems, followed by a single heterogeneous test to facilitate discussion of our preconditioning methods and detail motivation for the development of additional hybrid approaches. A parametric study then compares all of our hybrid methods to the individual methods of which they are comprised as well as traditional acceleration methods. This study is used to determine the general impact of problem parameters on the performance of our hybrid methods. Realistic test problem configurations containing void regions then confirm that the findings of our controlled testing configurations extend to realistic geometries. Finally, we provide numerical results illustrating the construction cost associated with algorithms for constructing the *ITMM* matrices, including our newly developed algorithm.

Finally, in Chapter 5 we test both *PBJ-ITMM* and *IPBJ* in parallel execution on unstructured tetrahedral grids using our developed algorithm for construction of the *ITMM* matrices. These tests, utilizing up to 32,768 processors, quantify the scaling in the amount of time required to construct the *ITMM* matrices and subsequently solve the transport equation iteratively for multiple problems with varying amounts of parallelization. We then perform tests on a problem

containing a void region, demonstrating the iterative slowdown our hybrid methods have been demonstrated to alleviate in Cartesian grids.

After presentation of our analysis and results, conclusions are reached and potential future work is described in Chapter 6.

## 1.5: Terminology

**Robustness:** The term “robustness” in the context of iterative methods refers to the existence of a finite upper bound on the number of iterations required by an iterative method to converge to a given criterion. We describe a method as robust with respect to a given parameter if, in an infinite homogeneous medium, the method satisfies this requirement for all values of the specified parameter over its entire possible range.

**Hybrid method abbreviations:** In this work, we develop, analyze, and test multiple hybrid methods. These methods utilize a secondary sweep-based method to accelerate the convergence of a primary *PBJ*-type method. There are two approaches for the hybrid combination of the two methods, Preconditioning (*P*) and *Asynchronous Hybrid* (*AH*). The differences between these two approaches will be discussed at length during their development, however, to summarize, *P* indicates both methods are executed globally in staggered manner, whereas *AH* indicates that a given sub-region of a problem has only one of the two iteration methods applied to it. We abbreviate the hybrid methods with a three-part acronym, separated by hyphens. The first letter indicates the manner in which the two methods are combined (*P* or *AH*), the second indicates the primary method (*PI* for *PBJ-ITMM* or *IP* for *IPBJ*), and the third indicates the secondary method (*SI* or *IP*). For example, the asynchronous hybrid combination of *PBJ-ITMM* and *SI* is abbreviated as *AH-PI-SI*.

**Degree of parallelism:** Degree of parallelism is a term we use to describe the degree to which a hybrid method reduces the effectiveness of parallel execution in absence of communication costs compared to the primary method. Neglecting communication time, if a *PBJ* method executes in parallel with one sub-domain per processor and all sub-domains comprised of the same number of cells, all sub-domains will finish execution at the same time and the next iteration can begin

immediately. With hybrid methods, though, the secondary method may cause a delay between the completion of the *PBJ* sub-domain solution and the point at which the subsequent iteration may begin. The length of this delay indicates the penalty to the degree of parallelism imposed by the hybrid method, as this delay effectively increases the pre-iteration execution time.

Unstructured grid (mesh): A spatial mesh of non-Cartesian cells with no prescribed patterning.

Convergence rate or rate of convergence: The reduction of the iterative error due to the application of a single iteration. A method with a higher convergence rate converges in fewer iterations. This term does not imply anything about the relative execution time of a method's iterations.

# Chapter 2: Literature Review

## 2.1: Acceleration of Neutron Transport Methods

Since iterative methods are implemented to avoid infeasibly long runtimes in the process of obtaining transport solutions, much work has been done to assess and improve their convergence rates. As this work is concerned with accelerating the inner iterations, we will focus on the techniques which have previously been used to accelerate these iterations, as opposed to the outer iterations. First, we discuss the technique researchers have used to assess the effectiveness of the unaccelerated *SI* method.

This analysis employs a Fourier transformation, projecting the iterative error (the “distance” between the converged solution and a given iterate of the scalar flux) onto a Fourier basis. As Fourier functions are periodic in nature, the analysis is conducted on a problem whose solution is periodic in space. This implies that this analysis must be performed on an infinite medium. Additionally, while the analysis has most frequently been performed in homogeneous media, the analysis only strictly requires that the medium’s configuration (geometry and nuclear data) be periodic in space. With the iterative error projected onto a Fourier function space (transforming from the spatial domain to the frequency domain), the scaling of the error from one iteration to the next can be derived as a function of the Fourier variable(s). These scaling factors are the eigenvalues of the *SI* transport operator, scaling their associated eigenfunctions with the execution of a single *SI* iteration.

Conducting this Fourier analysis on the *SI* iterative method with one energy group yielded a spectral radius (the largest magnitude eigenvalue) of this method that is equal to the medium’s scattering ratio in an infinite homogeneous medium. [7] This indicates that as the scattering ratio approaches unity, the spectral radius also approaches unity, making the number of required iterations grow without bound for an infinite medium. This maximum magnitude eigenvalue occurs at the origin of the Fourier space. This result can also be explained physically. It is possible to show that the solution to the  $s^{th}$  iteration is the flux of particles which have undergone up to  $s-1$  collisions. Therefore, configurations in which particles, on average, survive many collisions will

take more iterations to converge. [6, 7] In the following sections, we will discuss several acceleration methods developed to increase the rate of convergence in these problems. During the discussion, we will provide equations from the literature when appropriate, and we will provide the basic equations used to implement the method for a sample problem type as well as any equations necessary to demonstrate specific qualities of the resulting iterative scheme. Note that while we modify the notation in these equations to match the notation adopted in the remainder of this dissertation, some methods may use the same symbols for auxiliary terms as defined in the source reference.

### ***2.1.1: Diffusion Synthetic Acceleration (DSA)***

Accelerating the *SI* method in problems with scattering ratios close to unity has been an active area of research. A precursor to some of the earliest work in the field of *SI* acceleration was done by *Kopp* [8], who demonstrated that the transport solution could be obtained by the synthetic method, a method which poses the solution as the superposition of multiple “parts” of the solution. While *Kopp*’s work used the synthetic method to solve rather than to accelerate, his work provided the foundation for early attempts to accelerate transport iterations by introducing a low order operator which would solve the components of the transport solution which were observed to converge slowly. *Reed* [9] applied this idea of solution synthesis to an iterative process, equating the current iterate of the scalar flux to the scalar flux obtained by the mesh sweep, plus an additional term that satisfies a diffusion-like equation driven by a distributed source proportional to the iterative change in the scalar flux. This yields the system of equations,

$$\phi^{(s+\frac{1}{2})} = \mathbf{K}(\boldsymbol{\Sigma}_s \phi^{(s)} + q), \quad (2.1)$$

$$r^{(s+\frac{1}{2})} = \phi^{(s+\frac{1}{2})} - \phi^{(s)}, \quad (2.2)$$

and

$$\phi^{(s+1)} = \phi^{(s+\frac{1}{2})} + (\mathbf{D} - \boldsymbol{\Sigma}_s)^{-1} \boldsymbol{\Sigma}_s r^{(s+\frac{1}{2})}. \quad (2.3)$$

In these equations,  $s$  is the iteration index,  $\mathbf{D}$  is the matrix representation of a diffusive operator,  $\mathbf{\Sigma}_s$  is a diagonal matrix of scattering cross sections, and  $\mathbf{K}$  is the transport matrix. These square matrix operators have dimensions equal to the number of spatial degrees of freedom. When using an iterative method which utilizes a two-step method of any kind, we use  $s$  to indicate quantities before the iteration,  $s + \frac{1}{2}$  to indicate quantities resulting from the first step (usually the transport sweep), and  $s+1$  to indicate quantities after the second step, whatever that second step may be (usually a low order solution). This notation will be used repeatedly throughout this report.

Equation (2.1) is the *SI* iterative solution in matrix notation, making the first right-hand-side term of Eq. (2.3) the scalar flux resulting from the mesh sweep. This is then added to the second right-hand-side term, which is a diffusive matrix multiplied by the iterative change in the scalar flux,  $r$ , synthesizing the updated iterative scalar flux. *Reed* performs analysis on this synthetic method to obtain conditions the employed diffusive operator must meet to ensure stability of the synthetic method. Numerical testing of this method showed a large reduction in the number of required iterations over the unaccelerated problem, but also showed instability (lack of convergence) in problems when the diffusion coefficient was small relative to the size of the spatial cells.

The synthetic method was later used by *Gelbard* and *Hageman* [10] to form the *Diffusion Synthetic Acceleration (DSA)* method, which has since become one of the most widely used accelerators for transport calculations. *DSA* employs a preconditioner technique which will be used by many of the other acceleration methods covered by this review. [6] In general, a preconditioner is a mathematical, typically low-order, operator inserted within the sequence of an iterative scheme to accelerate its convergence. This low-order operator is an approximation of the transport operator, with the approximations made in a manner such that the resulting matrix is far cheaper to factor and solve than the matrix representing the full transport operator. In *DSA*, this preconditioner matrix is obtained by use of the diffusion approximation, which is derived by assuming an angular flux with linear angular anisotropy. This approach is used because the eigenmodes of the *SI* operator which are slowest to converge are those with frequency close to zero, the so-called “flat” modes. These modes represent the portion of the solution with a nearly linear dependence of the angular flux on direction, therefore causing diffusion theory to be an accurate approximation in that region of Fourier space. [7] The *DSA* acceleration method applies the diffusion operator to the stage of computing the iterative residual, thus ensuring that the

converged solution will not be altered by the acceleration process and will match the unaccelerated limit to within the convergence criterion. This follows from the observation that the acceleration method will cease to impact the iterative solution as the iterative residual approaches zero with iterative convergence.

Fourier analysis of the *SI* method accelerated with *DSA* in slab geometry established that the spectral radius is bounded from above by  $0.23c$ , where  $c$  is the scattering ratio. This implies that for highly scattering problems, which were slow to converge using the unaccelerated *SI* method, the number of required iterations will be much smaller when *DSA* is used. Additionally, the analysis establishes that *DSA* is robust, since the spectral radius is bounded below unity, implying that the number of iterations required to achieve a fixed convergence criterion cannot grow unboundedly as the problem size increases and  $c \rightarrow 1$ . [10]

While most of the computational results reported in the literature support this finding, instability of *DSA*-accelerated *SI* was observed in problems with optically thick cells, with many of these problems not converging when *DSA* acceleration was used. *Alcouffe* conjectured this to be the result of inconsistency between the discretization of the high-order transport operator and the low-order *DSA* operator. This explained why the instability only occurred in computational tests, as the theoretical analysis was conducted using the continuum operators. The Fourier analysis was therefore not subject to inconsistencies between the discretizations of the transport and diffusion operators. In previous works, the starting point for implementing *DSA* was a continuum diffusion-like low-order equation, followed by the application of a selected discretization to obtain the linear algebraic system of low-order equations. *Alcouffe* demonstrated that this resulted in inconsistency between the high-order and low-order discretizations, which ultimately caused the observed instability. *Alcouffe* derived a new *DSA* formulation which began with the discretized transport equation, then applied the diffusion approximation by projection onto Legendre polynomials of orders less than or equal to one. The resulting acceleration method was shown to be stable for all cell thicknesses when used with the *DD* spatial discretization method in one-dimensional slab geometry. [11]

*Alcouffe*'s work suggested that the *DSA* derivation must start with the discretized transport equation, indicating that the low-order *DSA* equations must be derived specifically for the particular spatial discretization method of the transport equation that they are to be used with. This motivated work by *Larsen* to derive unconditionally stable *DSA* equations in slab geometry for a

variety of  $S_N$  spatial discretization methods and to formalize the derivation process to aid in future derivations; the resulting process he termed the “four step method.” [7] Larsen’s unconditionally stable *DSA* method based on the four-step approach was extended by *Azmy* to two-dimensional Cartesian geometry for the *AHOT-N0* method. [12, 13] Testing of the method showed it to greatly reduce the number of iterations required for one- and two-dimensional problems, in both homogeneous and heterogeneous configurations, when used with *AHOT-N0*. The number of required iterations was observed to be relatively insensitive to scattering ratio and cell optical thickness, and was found to be sensitive to quadrature order only at extremely low orders. *Azmy* and *Larsen* [14] conducted a Fourier analysis of this method and showed its spectral radius to be close to 0.25 for all cases, thus providing theoretical support for the aforementioned variation in the number of required iterations being so small.

We provide the following equations, originally derived by *Azmy*, [12, 13] which demonstrate the implementation of *DSA* in two-dimensional geometry using the *AHOT-N0* spatial discretization method.

First, the following two equations are solved for all  $f_{00}$  terms, the zeroth order Legendre angular moments of the scalar flux residual in both dimensions.

$$\begin{aligned}
& [\Theta_{k,j} \Sigma_{Rx,k,j} - D_{x,k,j} + \Theta_{k+1,j} \Sigma_{Rx,k+1,j} - D_{x,k+1,j}] f_{00,k,j}^{x+,(s+1)} \\
& + [\Theta_{k,j} \Sigma_{Rx,k,j} + D_{x,k,j}] f_{00,k,j}^{x-,(s+1)} \\
& + [\Theta_{k+1,j} \Sigma_{Rx,k+1,j} + D_{x,k+1,j}] f_{00,k+1,j}^{x+,(s+1)} \\
& - \Theta_{x,k,j} \Sigma_{Ry,k,j} [f_{00,k,j}^{y+,(s+1)} + f_{00,k,j}^{y-,(s+1)}] \\
& - \Theta_{x,k+1,j} \Sigma_{Ry,k+1,j} [f_{00,k+1,j}^{y+,(s+1)} + f_{00,k+1,j}^{y-,(s+1)}] \\
& = \Theta_{k,j} g_{x,k,j}^{(l+1)} - \Theta_{x,k,j} g_{y,k,j}^{(s+1)} + \Theta_{k+1,j} g_{x,k+1,j}^{(s+1)} - \Theta_{x,k+1,j} g_{y,k+1,j}^{(s+1)}
\end{aligned} \tag{2.4}$$

$$\begin{aligned}
& [\Theta_{k,j} \Sigma_{Ry,k,j} - D_{y,k,j} + \Theta_{k,j+1} \Sigma_{Ry,k,j+1} - D_{y,k,j+1}] f_{00,k,j}^{y+,(s+1)} \\
& + [\Theta_{k,j} \Sigma_{Ry,k,j} + D_{y,k,j}] f_{00,k,j}^{y-,(s+1)} \\
& + [\Theta_{k,j+1} \Sigma_{Ry,k,j+1} + D_{y,k,j+1}] f_{00,k,j+1}^{y+,(s+1)} \\
& - \Theta_{y,k,j} \Sigma_{Rx,k,j} [f_{00,k,j}^{x+,(s+1)} + f_{00,k,j}^{x-,(s+1)}] \\
& - \Theta_{y,k,j+1} \Sigma_{Rx,k,j+1} [f_{00,k,j+1}^{x+,(s+1)} + f_{00,k,j+1}^{x-,(s+1)}] \\
& = \Theta_{k,j} g_{y,k,j}^{(l+1)} - \Theta_{y,k,j} g_{x,k,j}^{(s+1)} + \Theta_{k,j+1} g_{y,k,j+1}^{(s+1)} - \Theta_{y,k,j+1} g_{x,k,j+1}^{(s+1)}
\end{aligned} \tag{2.5}$$

These equations utilize the following terms, [12, 13]

$$\rho_u \equiv \sum_{m=1}^M w_m \Omega_{u,m} \alpha_{u,m}, \quad u = x \text{ or } y \tag{2.6}$$

$$\gamma_{u,m} \equiv \alpha_{u,m} - 3\rho_u \Omega_{u,m}, \quad u = x \text{ or } y \tag{2.7}$$

$$\chi_{u,m} \equiv \Omega_{u,m} \alpha_{u,m} - \rho_u, \quad u = x \text{ or } y \tag{2.8}$$

$$\theta_u \equiv \frac{\frac{1}{\Delta v}}{\frac{1}{\Delta u} + \frac{3\rho_u}{2} (\Sigma_t - \Sigma_s)}, \quad v \neq u = x \text{ or } y \tag{2.9}$$

$$\Sigma_{Ru} \equiv \frac{\frac{\Sigma_t - \Sigma_s}{2}}{\frac{1}{\Delta v} + \frac{3\rho_u}{2} (\Sigma_t - \Sigma_s)}, \quad v \neq u = x \text{ or } y \tag{2.10}$$

$$D_u \equiv \frac{1}{\frac{3\Delta u}{2} \Sigma_t} + \rho_u, \quad u = x \text{ or } y \tag{2.11}$$

$$g_{u,k,j}^{(s+1)} \equiv \frac{\Sigma_s \left( \phi_{k,j}^{(s+\frac{1}{2})} - \phi_{k,j}^{(s)} \right)}{\frac{1}{\Delta u} + \frac{3\rho_u}{2} (\Sigma_t - \Sigma_s)}, \quad u = x \text{ or } y \quad (2.12)$$

$$\Theta \equiv \frac{1}{\theta_x \theta_y - 1} \quad (2.13)$$

$$\Theta_u \equiv \theta_u \Theta, \quad u = x \text{ or } y \quad (2.14)$$

In these equations, the subscripts  $k$  and  $j$  indicate the spatial cell  $(k,j)$  on a Cartesian grid,  $a_{u,m}$  are the  $u$  dimension's *AHOT-N0* spatial weighting factors for direction  $m$ , and  $\Delta x$  and  $\Delta y$  are the size of spatial cell  $(k,j)$  in the  $x$  and  $y$  directions, respectively. Finally,  $f_{pq,k,j}$  is the  $p^{\text{th}}$   $x$  direction /  $q^{\text{th}}$   $y$  direction Legendre angular moment of the angular flux residual (converged solution minus current value) in cell  $(k,j)$ . The superscript  $u\pm$  refers to the quantity on the  $x/y$  cell face with the  $\pm$  distinguishing between the right/left and top/bottom faces when  $u = x$  and  $y$ , respectively.

We refer to Eqs. (2.4) and (2.5) as the  $x$  and  $y$  direction *DSA* equations, as they couple the edges of a cell to those of its neighbors to the right and top, respectively. When supplemented with the appropriate boundary conditions, supplied by *Azmy* [13], Eqs. (2.4) and (2.5) form the *DSA* matrix which is factored as a pre-process, then solved after each transport mesh sweep to accelerate convergence. From these two equations we are able to see the stencil of the *DSA* matrix that is consistent with *AHOT-N0* to be a 7-stripped matrix for 2-D, as there are a total of seven unique  $f_{00}$  terms in each of these equations, coupling the edges of two cells in the aforementioned pattern.

This factored matrix equation is solved for  $f_{00}$  at each cell edge within the problem domain. With these values known, *Azmy* then provides the following equations which are used to solve for  $f_{10}$  and  $f_{01}$ , the first Legendre moments of the angular flux residual, at each cell edge. Recall that neglecting all Legendre angular moments of higher orders (including bilinear moments) is the primary approximation of the low order *DSA* equations.

$$\begin{aligned}
f_{10,k,j}^{x+,(s+1)} &= \frac{1}{2} (\Theta_{k,j} \Sigma_{Rx,k,j} - D_{x,k,j}) f_{00,k,j}^{x+,(s+1)} + \frac{1}{2} (\Theta_{k,j} \Sigma_{Rx,k,j} + D_{x,k,j}) f_{00,k,j}^{x-,(s+1)} \\
&\quad - \frac{1}{2} \Theta_{x,k,j} \Sigma_{Ry,k,j} [f_{00,k,j}^{y+,(s+1)} + f_{00,k,j}^{y-,(s+1)}] \\
&\quad + \frac{1}{2} (\Theta_{x,k,j} g_{y,k,j}^{(s+1)} - \Theta_{k,j} g_{x,k,j}^{(s+1)})
\end{aligned} \tag{2.15}$$

$$\begin{aligned}
f_{10,k,j}^{x-,(s+1)} &= \frac{1}{2} \Theta_{x,k,j} \Sigma_{Ry,k,j} [f_{00,k,j}^{y+,(s+1)} + f_{00,k,j}^{y-,(s+1)}] - \frac{1}{2} (\Theta_{k,j} \Sigma_{Rx,k,j} + D_{x,k,j}) f_{00,k,j}^{x+,(s+1)} \\
&\quad - \frac{1}{2} (\Theta_{k,j} \Sigma_{Rx,k,j} - D_{x,k,j}) f_{00,k,j}^{x-,(s+1)} - \frac{1}{2} (\Theta_{x,k,j} g_{y,k,j}^{(s+1)} - \Theta_{k,j} g_{x,k,j}^{(s+1)})
\end{aligned} \tag{2.16}$$

$$\begin{aligned}
f_{01,k,j}^{y+,(s+1)} &= \frac{1}{2} (\Theta_{k,j} \Sigma_{Ry,k,j} - D_{y,k,j}) f_{00,k,j}^{y+,(s+1)} + \frac{1}{2} (\Theta_{k,j} \Sigma_{Ry,k,j} + D_{y,k,j}) f_{00,k,j}^{y-,(s+1)} \\
&\quad - \frac{1}{2} \Theta_{y,k,j} \Sigma_{Rx,k,j} [f_{00,k,j}^{x+,(s+1)} + f_{00,k,j}^{x-,(s+1)}] \\
&\quad + \frac{1}{2} (\Theta_{y,k,j} g_{x,k,j}^{(s+1)} - \Theta_{k,j} g_{y,k,j}^{(s+1)})
\end{aligned} \tag{2.17}$$

$$\begin{aligned}
f_{01,k,j}^{y-,(s+1)} &= \frac{1}{2} \Theta_{y,k,j} \Sigma_{Rx,k,j} [f_{00,k,j}^{x+,(s+1)} + f_{00,k,j}^{x-,(s+1)}] - \frac{1}{2} (\Theta_{k,j} \Sigma_{Ry,k,j} + D_{y,k,j}) f_{00,k,j}^{y+,(s+1)} \\
&\quad - \frac{1}{2} (\Theta_{k,j} \Sigma_{Ry,k,j} - D_{y,k,j}) f_{00,k,j}^{y-,(s+1)} - \frac{1}{2} (\Theta_{y,k,j} g_{x,k,j}^{(s+1)} - \Theta_{k,j} g_{y,k,j}^{(s+1)})
\end{aligned} \tag{2.18}$$

With the zeroth and first Legendre moments obtained, Azmy's derivation then provides the following equations for updating the cell-averaged scalar flux obtained by the *SI* mesh sweep using these moments.

$$\phi_{k,j}^{(s+1)} - \phi_{k,j}^{(s+\frac{1}{2})} = \frac{1}{2} [f_{00,k,j}^{x+,(s+1)} + f_{00,k,j}^{x-,(s+1)}] + \frac{3\rho_x}{2} [f_{10,k,j}^{x+,(s+1)} - f_{10,k,j}^{x-,(s+1)}] \tag{2.19}$$

$$\phi_{k,j}^{(s+1)} - \phi_{k,j}^{(s+\frac{1}{2})} = \frac{1}{2} [f_{00,k,j}^{y+,(s+1)} + f_{00,k,j}^{y-,(s+1)}] + \frac{3\rho_y}{2} [f_{01,k,j}^{y+,(s+1)} - f_{01,k,j}^{y-,(s+1)}] \tag{2.20}$$

These equations comprise the *DSA* scheme for the *AHOT-N0* method in two-dimensional geometry and provide a significant reduction in the number of required iterations for convergence when compared to the unaccelerated *SI* method. We make note of a trait which is readily apparent in these equations; that the accelerated method converges to the same solution as unaccelerated *SI*. In Eq. (2.12), we clearly see that as the solution approaches convergence (and the iterative change in the scalar flux approaches zero), the  $g$  terms approach zero. We then see that this makes the right-hand-sides of Eqs. (2.4) and (2.5) approach zero, hence making all values of  $f_{00}$  equal to zero (assuming the *DSA* matrix operator is non-singular). Equations (2.15) - (2.18) then clearly produce zero values for all first moment scalar flux residuals. With zeroth and first moment scalar flux residuals equal to zero, the right-hand-sides of Eqs. (2.19) and (2.20) vanish, indicating that  $\phi_{k,j}^{(s+1)} = \phi_{k,j}^{(s+\frac{1}{2})}$ . This shows that upon convergence, the *DSA* acceleration step in the iterative sequence ceases to change the scalar flux, ensuring that the method converges to the same solution as the unaccelerated method.

Since the work on achieving consistency between the transport and diffusive discretizations, *DSA* has become one of the most widely used acceleration methods in transport calculations, and hence, research has continued to improve the method and to implement it for a wider variety of problems. Ideally, the scalar flux after a transport sweep will always be greater than the scalar flux preceding the sweep, resulting in a scalar flux iterative residual which is always positive. Mathematically, this corresponds to the following equation having an  $r$  value of less than unity; this  $r$  value resulting from a specific problem's discretization. [15]

$$\delta\phi^{(s+1)} = \phi^{(s+1)} - \phi^{(s)} = (1 - r)(\phi^{(\infty)} - \phi^{(s)}) \quad (2.21)$$

In realistic problems, this ideal progression to iterative convergence is often not the case, and the resulting negative scalar flux residuals hinder iterative stability of the *DSA* method, as this residual is used in the source term of the *DSA* equation. *Yamamoto* [15] has developed an algorithm for solving the *DSA* equations which mitigates this undesirable feature. This algorithm is called plus and minus separation and it works by executing two *DSA* solutions within each iteration, one for the positive scalar flux residuals obtained by the transport sweep and one for the negative. For the positive *DSA* problem, all negative residuals are set to zero and for the negative *DSA* problem, all

positive residuals are set to zero and the sign of the negative residuals is flipped. The *DSA* problem is then solved for each of these flux vectors and the *DSA* solution is the solution to the positive problem minus the solution to the negative problem. In numerical testing, *Yamamoto* found plus and minus separation to eliminate large spikes in magnitude of the iterative error that were otherwise observed during the process of converging a problem. This resulted in a large enough reduction in the number of required iterations to merit the increased cost of the *DSA* step, basically a doubling of its execution time.

More recently, work has been performed to extend the use of *DSA* to unstructured tetrahedral spatial meshes. *Muhammad* and *Hong* [16] have successfully implemented *DSA* acceleration in such meshes for a first order finite element spatial discretization. These authors implemented the *DSA* method into unstructured grids alongside the *Linear Discontinuous Expansion Method with Subcell Balance* method (*LDEM-SCB*). *LDEM-SCB* employs a discretization which divides a single tetrahedron into multiple sub-cells. For consistency, the authors imposed the same discretization for the *DSA* equations. The resulting *DSA* equations are solved iteratively, thus creating a nested iterative sequence in which the solution of the low order problem requires iterations to solve. The authors use the *Linear Fine Mesh Rebalance* (*FMR*) method to accelerate these *DSA* iterations. When tested in a three-dimensional problem with a scattering ratio of 0.9999 (an extremely slowly converging problem for the *SI* method), this *DSA* with *FMR* approach was observed in the most dramatic case to reduce the runtime of a problem from over a day to just over an hour.

Additionally, *DSA* has been used by *Févotte* [17] to derive an acceleration method for 3-D Cartesian grids called *Piecewise Diffusion Synthetic Acceleration* (*PDSA*). *PDSA* is a diffusion-based accelerator constructed in a manner which allows for parallel solution. For the low order problem, *PDSA* divides the spatial domain into multiple sub-domains. Each of these sub-domains then undergoes two local *DSA* steps which differ by their local sub-domain boundary conditions. The first step (Neumann step) implements homogeneous Neumann boundary conditions, using the scalar fluxes obtained from the *SI* step, on all sub-domain boundaries which are not global problem boundaries. The second step (Dirichlet step) uses Dirichlet boundary conditions. The *DSA* equations are solved locally over each sub-domain in the Neumann step. The solution obtained from the Neumann step is then used to compute inhomogeneous Dirichlet boundary conditions for the ensuing Dirichlet step. These Dirichlet boundary conditions employ continuity of the scalar

flux at sub-domain interfaces by setting the scalar flux residual at an interface to the arithmetic average of the solutions on that face obtained in the Neumann step in neighboring sub-domains. The Dirichlet step then proceeds by solving the same *DSA* equations as in the Neumann step, but with different boundary conditions. This two-step process can be thought of as performing *DSA* independently in all sub-domains, each “unaware” of the others, then solving a second time with all sub-domains still solved independently, but with the condition that the *DSA* solution will be continuous across sub-domain boundaries.

Unlike other acceleration methods, the piecewise nature of *PDSA* makes the explicit derivation of the spectral radius difficult, if not impossible. Rather, Fourier analysis of the method was only able to produce an upper bound on the scaling of the iterative error; a bound which is dependent on the number of sub-domains used for the *PDSA* solution. Numerical testing of the *PDSA* acceleration method on a one-dimensional problem using the *DD* spatial discretization method demonstrated the method to have convergence properties identical to those of traditional *DSA* when the problem domain is split into only two sub-domains. This demonstrates the concept of *PDSA* using the Neumann step solutions to impose the coupling of adjacent sub-domains onto the Dirichlet step’s boundary conditions, as this would provide exact boundary conditions when all sub-domains of the problem are bordering each other. When the number of sub-domains was increased, however, the scaling of the iterative error grew until, when 9 sub-domains were used, the method became unstable.

### ***2.1.2: Adjacent-Cell Preconditioning (AP)***

While *DSA* was shown to provide acceleration for the *SI* iterative method, one observed shortcoming was the fact that it was formulated around cell-edge flux quantities rather than cell-averaged quantities. This results in a matrix of larger rank due to the number of edges in a problem being larger than the number of cells. Additionally, the resulting matrix may be less sparse since there is coupling between a cell’s edges and its neighbor’s edges, resulting in a larger number of non-zero matrix elements per row. For example, with *DSA* a 2-D problem results in a 7-diagonal matrix instead of a 5-diagonal matrix that would result in the low order matrix operator that couples a cell’s averaged quantities and the cell-averaged quantities of its four neighboring cells in the same 2-D problem configuration. Desire for an acceleration method which uses cell-averaged

quantities led to the development of *Imposed Diffusion Synthetic Acceleration (IDSA)*, a method similar to *DSA* that imposes cell-averaged flux coupling by neglecting the aforementioned principles of consistency between discretizations with the *DSA* method for the purpose of accelerating using the standard cell-centered diffusion template, rather than a consistent form derived from the correspondingly discretized transport equation. [9, 18]

While *IDSA* has not become widely used as an acceleration scheme, it was a precursor to the *Adjacent-Cell Preconditioning (AP)* method. In addition to being a viable acceleration method, studies of the *AP* acceleration method have demonstrated fundamental concepts which we utilize in our work.

The *AP* acceleration method employs the general concept of preconditioning in that it provides an approximation for the exact transport matrix that is significantly simpler and quicker to solve by inversion, factorization, or iteration than its exact counterpart. *AP* achieves this simplification by coupling the cell-centered scalar flux of a given spatial cell to the cell-centered scalar fluxes of all axially adjacent, not diagonally adjacent, cells, leading to a 3, 5, and 7 banded matrix for 1-D, 2-D, and 3-D Cartesian geometry problems, respectively. [19, 20] In these publications, *Azmy* derives the *AP* acceleration for one-dimensional and multi-dimensional problems, respectively. While we review the latter here, the general process is similar, regardless of dimensionality. The *AP* method formulation begins with the following update formula which provides the updated iterate of the scalar flux,  $\phi^{(s+1)}$ , by applying the preconditioner matrix,  $\mathbf{D}$ , to the difference between the scalar flux before and after the transport sweep,

$$\phi^{(s+1)} = \phi^{(s')} + \mathbf{D}^{-1} \left( \phi^{(s+\frac{1}{2})} - \phi^{(s)} \right). \quad (2.22)$$

As will be discussed shortly, the *AP* method is implemented differently depending on the optical thickness of the problem's cells. In this equation,  $\phi^{(s')} = \phi^{(s+\frac{1}{2})}$ , or  $\phi^{(s)}$  for optically thin or thick cells, respectively. The effectiveness of *AP* is contingent upon developing a prescription for the preconditioner matrix which is simple to solve and provides sufficient acceleration.

*Azmy* begins the development of such a preconditioner matrix with the observation that the performance of preconditioner matrices can vary drastically depending on the optical thickness of a problem's cells. The *AP* method therefore dictates the mathematical expression for the elements

of the preconditioner matrix in terms of cell geometric and nuclear properties differently depending on whether the cells are optically thin or thick, the resulting methods being termed *NAP* or *KAP*, respectively.

For the development of *KAP*, *Azmy* notes that intuition suggests the exact transport matrix (which the preconditioner matrix is attempting to approximate) will see the magnitude of its elements greatly reduced as they become further from the diagonal element when the problem's cells are optically thick. This concept was later proven mathematically by *Rosa*, *Azmy*, and *Morel*, who demonstrated that the exact transport matrix approaches a tridiagonal matrix as the optical thickness of cells increases for 1-D slab geometry problems. [21] We make a special note of this property, as it aids in certain developments presented in this work.

While this property suggests that the ideal *KAP* method would be to simply use only the elements of the exact transport matrix corresponding to the coupling of a cell to its Cartesian-adjacent neighbors, analysis suggests otherwise, as this prescription does not guarantee acceleration of the iterative error's flat mode, the mode that is known to be responsible for the iterative slowdown of the *SI* iterative method. This is to say that while this formulation for the *KAP* prescription may produce a solution close to that of the high order transport problem, it fails to resolve the components of the solution which the *SI* method requires assistance with. Fourier analysis of the *AP* method for homogeneous media with a uniform mesh produces a stability condition which is expressed as,

$$D_d = 1 - c - 2 \sum_{u=1}^U D_{o,u}, \quad (2.23)$$

where  $D_d$  and  $D_{o,u}$  are the diagonal and off-diagonal elements of the preconditioning matrix, respectively,  $u$  is the dimension index,  $U$  is the number of spatial dimensions in the problem, and  $c$  is the scattering ratio.

*Azmy* identifies a formulation for the *KAP* prescription that satisfies this stability condition and ensures that the flat mode has an associated eigenvalue of zero. This prescription dictates *KAP* to have the same diagonal elements as the exact transport matrix. This designation combined with the stability condition given by Eq. (2.23) results in the following prescription for the off-diagonal

elements of the *KAP* preconditioner matrix, with the diagonal elements subsequently calculated with Eq. (2.23).

$$D_{o,u} = -\frac{c}{2} \sum_{m=1}^M \frac{\omega_m \gamma_{m,u}}{1 + \sum_{u'=1}^U \gamma_{m,u'}}, \quad u = 1, \dots, U, \quad (2.24)$$

where we define,

$$\gamma_{m,u} \equiv \frac{\epsilon_{m,u}}{1 + \alpha_{m,u}}, \quad m = 1, \dots, M; \quad u = 1, \dots, U, \quad (2.25)$$

and

$$\epsilon_{m,u} \equiv \frac{2|\Omega_{u,m}|}{\Sigma_t \Delta u}, \quad m = 1, \dots, M; \quad u = 1, \dots, U. \quad (2.26)$$

In these equations,  $\Omega_{u,m}$  is the  $u$  dimension's component of discrete ordinate  $m$  and  $\Delta u$  is the  $u$  dimensional thickness of the spatial cell.

Fourier analysis shows this *KAP* prescription to provide excellent acceleration when cells are optically thick, especially when the thickness exceeds five *mean free paths* (*mfps*). However, this *KAP* prescription's high frequency modes approach instability as the cell optical thickness tends towards zero, reiterating the need for a separate prescription (*NAP*) for optically thin cells.

A *NAP* prescription for optically thin cells is obtained using the same process used to obtain the *KAP* prescription. For thin cells, the stability condition is,

$$D_d = \frac{1}{c} - c - 2 \sum_{u=1}^U D_{o,u}. \quad (2.27)$$

Through Fourier analysis, *Azmy* selects the following prescription for the off-diagonal elements of the preconditioner matrix.

$$D_{o,u} = -\frac{1}{4} \sum_{m=1}^M w_m \epsilon_{m,u} (\alpha_{m,u} + \epsilon_{m,u}), \quad u = 1, \dots, U \quad (2.28)$$

*NAP*'s prescription for the diagonal elements then follows using the stability condition, Eq. (2.27).

*Azmy*'s *NAP* prescription satisfies the stability condition obtained through Fourier analysis and ensures a zero eigenvalue for the flat mode. However, contrary to *KAP*, the *NAP* spectrum shows rapid acceleration for optically thin cells. Additionally, in optically thick cells, *NAP* does not approach instability as *KAP* did in optically thin cells, but the achieved acceleration is far less effective than provided by *KAP* with growing cell thickness.

To complete the *AP* formulation, a mixing formula for heterogeneous configurations and nonuniform meshes, and boundary conditions must be derived. The development and spectral analysis of *KAP* and *NAP* reported by *Azmy* assumed infinite homogenous media with uniform mesh. This is not the case with realistic problems, however, providing an additional complication when adjacent cells have different properties, potentially even belonging to separate types of *AP* formulation (i.e. one requiring *KAP* while another requiring *NAP*). To address this issue, *Azmy* provides a mixing formula along with comparison of the spectra for *NAP* and *KAP* to provide insight as to which prescription must be employed in cells of a given thicknesses, thus completing the formulation of the *AP* acceleration method.

To apply the mixing formula, off-diagonal elements are calculated for each cell using Eqs. (2.24) and (2.28) for optically thick and thin cells, respectively. For the heterogeneous (or non-uniform mesh) case, these are not the final off-diagonal elements of the preconditioner matrix, they are the off-diagonal elements of a hypothetical homogeneous problem comprised of cells identical to cell  $(k, j)$ . With homogeneous off-diagonal elements for each cell calculated, the heterogeneous off-diagonal elements for cell  $(k, j)$  are calculated using the mixing formula,

$$(D_{o,u}^{het})_{(k,j),(k',j')} = \frac{2\Delta v_{k,j}}{\frac{\Delta u_{k,j}}{D_{u,k,j}^{AP}} + \frac{\Delta u_{k',j'}}{D_{u,k',j'}^{AP}}} \quad (2.29)$$

where,

$$D_{u,k,j}^{AP} \equiv \Sigma_{t,k,j} \Delta^2 u (D_{o,u})_{k,j} . \quad (2.30)$$

$(k', j')$  is a cell adjacent to  $(k, j)$  in the  $u$  dimension and  $(D_{o,u}^{het})_{(k,j),(k',j')}$  are the final off-diagonal elements for a heterogeneous problem. Note that in Cartesian geometry, there will be two separate off-diagonal elements in each dimension. The final diagonal elements for a heterogeneous problem are then calculated using modified forms of Eqs. (2.23) and (2.27) for thick and thin cells, respectively,

$$(D_d^{het})_{k,j} = 1 - c - \sum_{(k',j')} (D_{o,u}^{het})_{(k,j),(k',j')} \quad (2.31)$$

and

$$(D_d^{het})_{k,j} = \frac{1}{c} - c - \sum_{(k',j')} (D_{o,u}^{het})_{(k,j),(k',j')} . \quad (2.32)$$

In these final diagonal element equations for a heterogeneous problem, the summation over  $(k', j')$  indicates a summation of the (in the 2-D case) four  $(D_{o,u}^{het})_{k,j}$  values calculated for neighboring cells.

Numerical testing of the *AP* method shows it to greatly reduce the number of required iterations compared to the *SI* method. Additionally, *AP* is found to converge in the same, or fewer number of iterations than the consistent *DSA* acceleration method for the attempted test problems. In problems with optically very thick cells, *KAP* even converges in a single iteration. [19, 20]

While initially observed to be unconditionally stable, later analysis by *Azmy* proves that there exists no choice of preconditioning matrix elements which couples only axially-adjacent cells that is unconditionally stable and robust. The proof of this impossibility is carried out on a periodic horizontal interface problem, which is a 2-D problem with uniform-mesh and homogeneous-material cells in the  $x$ -direction, but with cells in the  $y$ -direction alternating between two optical thicknesses. This provides the possibility of sharp material discontinuities when the difference in

the y-direction layers' optical thicknesses is large. The Fourier analysis of such a problem shows that for any chosen preconditioner matrix, there exists a problem for which stability of the  $\lambda_x = \lambda_y = 0$  mode indicates instability of the  $\lambda_x = \frac{\pi}{2}, \lambda_y = 0$  mode, where  $\lambda_x$  and  $\lambda_y$  are the Fourier variables. While the theoretical analysis was performed for cell-centered preconditioners, numerical results suggest that this impossibility extends to cell-edge based diffusive stencils (adjacent cell coupling only) preconditioners such as consistent *DSA* formalisms. [22] This shortcoming was later observed for *DSA* and mitigated (for *DSA*) through the use of *Krylov* iterative methods. [23]

### 2.1.3: Coarse Mesh Rebalance

The *Coarse Mesh Rebalance (CMR)* method utilizes the fact that, upon convergence, the solution must satisfy the neutron balance equation, but that any iterate prior to convergence may break this balance condition. *CMR* accelerates the solution by ensuring that this neutron balance is satisfied on a potentially coarser spatial mesh. To implement *CMR*, the spatial cells of a problem are grouped into sub-domains (denoted with subscript  $n$ ), resulting in a coarser mesh. The neutron balance equation is then integrated over angle, then sub-domain volume. The resulting equation balances the incoming and outgoing neutron currents (first angular moment of the angular flux) of a sub-domain with the production and loss of neutrons within the sub-domain. The set of equations obtained by applying this procedure to each sub-domain results in a sparse matrix, the solution of which is used to update the solution iterate. In a one-dimensional spherical geometry problem, this amounts to solving the following equation for all correction terms,  $f$ , on a coarse mesh. [2]

$$a_{n,n-1}f_{n-1} + a_{nn}f_n + a_{n,n+1}f_{n+1} = \tilde{q}_n \quad (2.33)$$

In this equation, the  $a$  terms are defined by,

$$a_{nn} = \sum_{i \in n} V_i \Sigma_{r,i} \phi_i^{(s)} + A_{n-} J_{n-}^{-,(s)} + A_{n+} J_{n+}^{+,(s)} \quad (2.34)$$

$$a_{n,n-1} = -A_{n-}J_{n-}^{+,(s)} \quad (2.35)$$

$$a_{n,n+1} = -A_{n+}J_{n+}^{-,(s)} \quad (2.36)$$

And the coarse mesh source term is defined by,

$$\tilde{q}_n = \sum_{i \in n} V_i q_i \quad (2.37)$$

In these equations,  $V_i$  is the volume of cell  $i$ , where the subscript can either denote a fine mesh cell with  $i$  or a coarse mesh cell with  $n$ .  $\Sigma_r$  is the removal macroscopic cross section,  $A$  is the area of a cell face, and  $J^{\pm,(s)}$  is the positive or negative partial current. The subscript  $n \pm$  denotes a quantity on either the positive or negative face of a coarse mesh cell respectively. Upon solving the linear system of equations resulting from Eq. (2.33) globally for  $f_n$ , the scalar flux iterate is updated using,

$$\phi_{m,i}^{(s+1)} = f_n \phi_{m,i}^{(s)}, \quad V_i \in V_n. \quad (2.38)$$

With the *CMR* acceleration method, it is easy to see that the number of sub-domains used to develop the coarse mesh can drastically affect the method's performance. This fact is ultimately the largest shortcoming of *CMR* acceleration, as the number of cells per sub-domain can greatly affect the rate of convergence and can even make the method diverge in some cases. Additionally, while it seems intuitive that the larger the number of sub-domains is, i.e. the less-coarse the coarse mesh is, the faster convergence will be, *Cefus* and *Larsen* [24] have shown that this is not always the case. In their tests, the convergence rate of *CMR* is found to have strong dependence on the optical thickness of a problem's cells. Their tests and analysis show that there is a range of optical thickness over which *CMR* is most effective, but deviation too far in either direction from this range can cause divergence of the iterations. They also show that for a test case, a larger number of coarse mesh cells did not always correspond to faster convergence. *CMR* has the benefit over other acceleration methods that it generally results in a smaller matrix to be solved in the low order

problem due to the coarsening of the mesh. The price that is paid for this benefit is that the performance of the resulting acceleration method can be unpredictable and, at times, divergent. [2, 24]

For this reason, *CMR* is not as frequently used in modern transport solvers as other acceleration methods such as *DSA*. This fact notwithstanding, some research has continued on *CMR*. *Park* and *Cho* [25] developed a *CMR* acceleration scheme for the method of characteristics, a method frequently used in reactor assembly transport simulations because it is able to deal with severely heterogeneous and non-Cartesian geometries. These authors use the convenient geometry of a reactor assembly to make each fuel cell a single coarse mesh cell in two-dimensional geometry, thus alleviating the uncertainty of requiring a user to specify the coarse mesh and the possibility to inadvertently cause iterative instability. The resulting method was shown to yield dramatic iterative acceleration over the unaccelerated method for 2-D assembly-level calculations.

#### **2.1.4: Quasidiffusion (*QD*)**

The *Quasidiffusion (QD)* acceleration method is different from all other acceleration methods discussed so far in the sense that it does not necessarily converge to the same solution as the unaccelerated *SI* method. Other acceleration methods are derived with the specific prescription that the neutron balance equation and auxiliary equations must hold upon convergence. *QD* was not derived with this constraint, and hence, it can converge to a different limit than *SI*. [26, 6]

The *QD* acceleration method effectively converges the neutron diffusion equation, but with a more accurate diffusion coefficient calculated in each iteration from the high order transport iterate. Solving the high order transport problem produces a new iterate of the angular flux which is used to calculate *Eddington Factors* by dividing the second angular moment of the angular flux by its zeroth angular moment. These *Eddington Factors* are used as effective diffusion coefficients in the low-order stage of the iterative sequence, a diffusion calculation whose flux solution comprises the current iterate. While the *QD* method converges to a different solution from the unaccelerated transport problem, it also converges to a solution different from that of the equivalent standard diffusion problem because the *Eddington Factors* do not assume linear anisotropy of the angular flux like the standard diffusion coefficient does. [6, 26]

*Adams and Larsen* [6] provide equations for *QD* in one-dimensional geometry with one energy group, isotropic scattering, and isotropic source. The process begins with the standard mesh sweep to obtain the angular flux distribution,  $\psi^{(s+\frac{1}{2})}(x, \mu)$ , where  $s$  is the iteration index,  $x$  is the spatial variable, and  $\mu$  is the directional cosine. This angular flux distribution is then used to calculate the *Eddington Factors*, defined as,

$$E^{(s+\frac{1}{2})}(x) \equiv \frac{\int_{-1}^1 \mu^2 \psi^{(s+\frac{1}{2})}(x, \mu) d\mu}{\int_{-1}^1 \psi^{(s+\frac{1}{2})}(x, \mu) d\mu}. \quad (2.39)$$

The Eddington Factors are then used to determine the updated scalar flux using the following low-order equation, obtained by taking the zeroth and first Legendre angular moments of the transport equation and neglecting all higher order terms.

$$-\frac{d}{dx} \frac{1}{\Sigma_t(x)} \frac{d}{dx} E^{(s+\frac{1}{2})}(x) \phi^{(s+1)}(x) + \Sigma_a(x) \phi^{(s+1)}(x) = q(x) \quad (2.40)$$

Note that the equations we provide are in the continuous form, meaning they must be discretized before they can be implemented in computer codes. The defining characteristics of the *QD* method are evident in the continuous form though.

We see from Eqs. (2.39) and (2.40) how *QD* differs from other diffusion-based acceleration methods. Firstly, the first term on the left-hand-side of Eq. (2.40) contains the *Eddington Factors* and the total macroscopic cross section rather than the diffusion coefficient. Unlike the standard diffusion coefficient, the *Eddington Factors* are not subject to the approximations of diffusion theory and are not fixed across iterations rendering the iterative operator algebraically nonlinear. Secondly, for all other acceleration methods, it has been easy to see how the low order problem will cease to impact the iterative solution as convergence is approached. This is to say, for all other acceleration methods considered thus far, we established that as  $\phi^{(s+1)}$  approaches  $\phi^{(\infty)}$ ,  $\phi^{(s+1)} - \phi^{(s+\frac{1}{2})}$  approaches zero. For the *QD* equations, no such feature applies, indicating the aforementioned property that *QD* may, and typically does, converge to a solution which differs from that of the unaccelerated problem.

*Anistratov* and *Cornejo* [27, 28] have studied *QD* for use in reactor core  $k$ -eigenvalue calculations. These studies have shown a large reduction in the number of required iterations compared to unaccelerated *SI* while producing scalar flux and multiplication factors of comparable accuracy. Additionally, these authors have studied multilevel *QD* methods which implement the method on a sequence of coarsened energy grids. These multilevel methods obtained the *QD* solution to problems with a smaller number of low order solutions than the standard *QD* method, thus reducing the resulting method's computational cost.

The *QD* method has also been extended by *Wieselquist*, *Anistratov*, and *Morel* [29] to unstructured two-dimensional quadrilateral meshes. This method used short characteristics to discretize the transport equation and an advanced diffusion equation discretization for unstructured meshes. This method was tested on a model problem for which the analytical transport solution is known. These tests demonstrated second order convergence of both the scalar flux and current solutions obtained on randomized spatial meshes.

### ***2.1.5: Nonlinear Diffusion Acceleration (NDA)***

Another acceleration method which has become widely used for  $S_N$  transport methods is the *Nonlinear Diffusion Acceleration (NDA)* method. The underlying principle of *NDA* is similar to that of *DSA*; using a low order, diffusion-like operator to accelerate the transport iterations since diffusion theory rapidly converges the error modes with frequencies close to zero, as those modes have linear or nearly linear angular anisotropy of the solution. The development of *NDA* was motivated by the desire for an acceleration method that was not subject to the aforementioned loss of robustness in strongly heterogeneous problems that were observed with *AP* and *DSA*.

The *NDA* method was originally introduced by *Smith* and *Rhodes* [30]. These authors introduce the basic concept of accelerating the *SI* method using a nonlinear diffusive low order problem on a coarse mesh with a modified, nonlinear diffusion coefficient calculated using,

$$J_{n+\frac{1}{2}} = -D_{n+\frac{1}{2}}(\phi_{n-1} - \phi_n) - \tilde{D}_{n+\frac{1}{2}}(\phi_{n+1} + \phi_n). \quad (2.41)$$

In this equation,  $J$  is the neutron current (first angular moment of the angular flux),  $D$  is the diffusion coefficient,  $\tilde{D}$  is a modified diffusion coefficient, and  $n$  is the coarse mesh cell index,

with  $n + \frac{1}{2}$  denoting the cell edge between cells  $n$  and  $n+1$ . This equation is a modified form of Fick's Law, with a correction term appended to the right-hand-side. The fundamental concept of *NDA* is that if  $\tilde{D}$  were to be calculated using (2.41) and fully converged fluxes and currents, then this same equation can subsequently produce the fully converged current, thus eliminating the approximate nature of Fick's Law. Thus, the conjunction of Eq. (2.41) and a diffusion-like neutron balance equation provides, upon convergence, the same result as the discretized transport equation. This is referred to as closure between the high-order and low-order problems. *Smith* and *Rhodes* find *NDA* to accelerate the sharply heterogeneous environment of a fuel assembly effectively.

*Park*, *Knoll*, and *Newman* [31] and *Cornejo* and *Anistratov* [32] later tested *NDA* for use in  $k$ -eigenvalue problems. These authors provide algorithms for the implementation of *NDA* in  $k$ -eigenvalue problems and provide results showing acceleration comparable to that previously reported by *Smith* and *Rhodes*. More importantly for the purposes of our review, *Cornejo* and *Anistratov* [32] provide discretized *NDA* equations, which we modify to be for a fixed source problem rather than  $k$ -eigenvalue, and hence, suitable for implementation in our test code, HAT-2C. In 2-dimensional discrete meshes, the modified Fick's Law is formulated as,

$$J_{x,k+\frac{1}{2},j} = -\frac{D_{k+\frac{1}{2},j}(\phi_{k+1,j} - \phi_{k,j})}{\Delta x_{k+\frac{1}{2}}} + \frac{1}{2} \tilde{D}_{k+\frac{1}{2},j}(\phi_{k+1,j} + \phi_{k,j}) \quad (2.42)$$

and

$$J_{y,k,j+\frac{1}{2}} = -\frac{D_{k,j+\frac{1}{2}}(\phi_{k,j+1} - \phi_{k,j})}{\Delta y_{j+\frac{1}{2}}} + \frac{1}{2} \tilde{D}_{k,j+\frac{1}{2}}(\phi_{k,j+1} + \phi_{k,j}). \quad (2.43)$$

The indices  $k$  and  $j$  denote the spatial cell  $(k, j)$  with half indices indicating cell edges. The edge dimensions are calculated using,

$$\Delta x_{k+\frac{1}{2}} = \frac{\Delta x_{k+1} + \Delta x_k}{2} \quad \text{and} \quad \Delta y_{j+\frac{1}{2}} = \frac{\Delta y_{j+1} + \Delta y_j}{2}. \quad (2.44)$$

Upon completion of a single *SI* iteration, this prescription for a modified Fick's Law calculates the modified diffusion coefficients using,

$$\tilde{D}_{k+\frac{1}{2},j}^{(s+\frac{1}{2})} = \frac{J_{x,k+\frac{1}{2},j}^{(s+\frac{1}{2})} + \frac{D_{k+\frac{1}{2},j}}{\Delta x_{k+\frac{1}{2}}} \left( \phi_{k+1,j}^{(s+\frac{1}{2})} - \phi_{k,j}^{(s+\frac{1}{2})} \right)}{\frac{1}{2} \left( \phi_{k+1,j}^{(s+\frac{1}{2})} + \phi_{k,j}^{(s+\frac{1}{2})} \right)} \quad (2.45)$$

and

$$\tilde{D}_{k,j+\frac{1}{2}}^{(s+\frac{1}{2})} = \frac{J_{y,k,j+\frac{1}{2}}^{(s+\frac{1}{2})} + \frac{D_{k,j+\frac{1}{2}}}{\Delta y_{j+\frac{1}{2}}} \left( \phi_{k,j+1}^{(s+\frac{1}{2})} - \phi_{k,j}^{(s+\frac{1}{2})} \right)}{\frac{1}{2} \left( \phi_{k,j+1}^{(s+\frac{1}{2})} + \phi_{k,j}^{(s+\frac{1}{2})} \right)}. \quad (2.46)$$

As with previous methods,  $s$  is the iteration index, with  $s + \frac{1}{2}$  indicating a quantity's value after the *SI* sweep. Diffusion coefficients at cell edges are calculated using,

$$D_{k+\frac{1}{2},j} = \frac{2D_{k,j}D_{k+1,j}\Delta x_{k+\frac{1}{2}}}{D_{k,j}\Delta x_{k+1} + D_{k+1,j}\Delta x_k} \quad \text{and} \quad D_{k,j+\frac{1}{2}} = \frac{2D_{k,j}D_{k,j+1}\Delta y_{j+\frac{1}{2}}}{D_{k,j}\Delta y_{j+1} + D_{k,j+1}\Delta y_j}, \quad (2.47)$$

where the standard cell-centered diffusion coefficients are,

$$D_{k,j} = \frac{1}{3\Sigma_{t,k,j}}. \quad (2.48)$$

The neutron currents are calculated through the numerical integration,

$$J_{x,k+\frac{1}{2},j}^{(s+\frac{1}{2})} = \sum_{m=1}^M \Omega_{x,m} w_m \psi_{m,k+\frac{1}{2},j}^{(s+\frac{1}{2})} \quad \text{and} \quad J_{y,k,j+\frac{1}{2}}^{(s+\frac{1}{2})} = \sum_{m=1}^M \Omega_{y,m} w_m \psi_{m,k,j+\frac{1}{2}}^{(s+\frac{1}{2})}. \quad (2.49)$$

Unlike *AP* and *DSA* where the high order solution passes the scalar flux (or more precisely the iterative change in the scalar flux) to the low order problem, *NDA* passes the modified diffusion coefficients to the low order problem, which were obtained using the high order solution. The low order *NDA* solution is then the solution of the zeroth angular moment of the transport equation, which, for a 2-D, monoenergetic, steady-state, fixed source problem with isotopic scattering is,

$$\begin{aligned} & \left( J_{x,k+\frac{1}{2},j}^{(s+1)} - J_{x,k-\frac{1}{2},j}^{(s+1)} \right) \Delta y_j + \left( J_{y,k,j+\frac{1}{2}}^{(s+1)} - J_{y,k,j-\frac{1}{2}}^{(s+1)} \right) \Delta x_k + (\Sigma_{t,k,j} - \Sigma_{s,k,j}) \Delta x_k \Delta y_j \phi_{k,j}^{(s+1)} \\ & = \Delta x_k \Delta y_j q_{k,j}. \end{aligned} \quad (2.50)$$

This equation imposes no additional approximations, hence it possesses the same solution as the transport equation. It must be supplemented, however, with a relationship between the cell-edge current and the cell-averaged scalar flux. *NDA* provides this relation through the modified Fick's Law of Eqs. (2.42) and (2.43) as,

$$J_{x,k+\frac{1}{2},j}^{(s+1)} = - \frac{D_{k+\frac{1}{2},j} \left( \phi_{k+1,j}^{(s+1)} - \phi_{k,j}^{(s+1)} \right)}{\Delta x_{k+\frac{1}{2}}} + \frac{1}{2} \tilde{D}_{k+\frac{1}{2},j}^{(s+\frac{1}{2})} \left( \phi_{k+1,j}^{(s+1)} + \phi_{k,j}^{(s+1)} \right) \quad (2.51)$$

and

$$J_{y,k,j+\frac{1}{2}}^{(s+1)} = - \frac{D_{k,j+\frac{1}{2}} \left( \phi_{k,j+1}^{(s+1)} - \phi_{k,j}^{(s+1)} \right)}{\Delta y_{j+\frac{1}{2}}} + \frac{1}{2} \tilde{D}_{k,j+\frac{1}{2}}^{(s+\frac{1}{2})} \left( \phi_{k,j+1}^{(s+1)} + \phi_{k,j}^{(s+1)} \right). \quad (2.52)$$

The iterative sequence of *NDA* is therefore as follows. Execution of the *SI* sweeps yields the current and flux distributions which are then used to calculate modified diffusion coefficients with Eqs. (2.45) and (2.46). These modified diffusion coefficients are then used to solve Eqs. (2.50), (2.51), and (2.52), producing the scalar fluxes that are passed to the high-order phase of the subsequent iteration to repeat the process until convergence is observed. The *NDA* equations must also be supplemented by boundary conditions, which are available in [32].

The provided *NDA* equations illustrate a few key aspects of the method. Firstly, these equations demonstrate the closure of the high and low order problems, the defining feature of *NDA*. Equations (2.45) and (2.46) and Eqs. (2.51) and (2.52) are clearly forms of Eqs. (2.42) and (2.43), with the only difference being iteration indices. Thus, upon convergence, Eqs. (2.51) and (2.52) become exact, making the low order solution equivalent to that of the high order problem. Additionally, the nonlinearity of *NDA* requires that, unlike with linear methods, the *NDA* matrix must be reconstructed each iteration, as its elements will change. This is clear as the modified diffusion coefficients change each iteration, which appear in matrix elements.

*Anistratov* [33] also investigated multilevel *NDA* methods. These multilevel methods use a nested iteration sequence in which the low order problem itself involves an iterative process. There are three varieties of *NDA* studied, *Two Level NDA (TLNDA)*, *Multilevel NDA (MLNDA)*, and *Grey NDA (GNDA)*. In addition to the transport sweep which all of these methods use, there are two other steps involved. These steps are the multigroup low order iterations which solve the multigroup *NDA* equations, and Newton's iterations which solve the one-group *NDA* equations. *TLNDA* uses the former, *GNDA* uses the latter, and *MLNDA* uses both with the Newton iterations used to accelerate the solution of the multigroup *NDA* equations. *GNDA* is found to require almost nine times the number of transport sweeps as either of the other two. *TLNDA* and *MLNDA* require the same number of transport sweeps, which is expected as the Newton iterations in *MLNDA* are used simply to accelerate the low order solution, not to improve its accuracy. The Newton iterations in *MLNDA*, however, are found to greatly reduce the number of multigroup low order iterations required, improving the overall computational cost of the method.

*Xiao, Ren, and Wang* [34] have studied the combination of *NDA* and *CMR* into a method named *LR-NDA*. This method strives to improve the convergence rate as well as eliminate the unpredictable iterative behavior associated with *CMR*. *LR-NDA* accelerates by solving the low order *NDA* equations on a coarse mesh. In areas where cells are optically thick though, the coarse mesh cell is locally refined and the *NDA* equations are solved on this finer mesh, using local boundary conditions provided by the mesh sweep solution. The authors note that the added execution time associated with this refinement is very small, as the refined meshes utilize local boundary conditions, allowing for solution in regions in need of refinement to be computed in parallel. When tested on a problem with several regions of optically thick cells, *LR-NDA* was observed to have an improved convergence rate over *CMR*, and the potential iterative instabilities

associated with *CMR* were eliminated. This comparison to *CMR* demonstrates the ability of *LR-NDA* to accelerate a problem using a mesh which is, in many regions of the problem, of coarseness associated with *CMR*, but without experiencing the iterative instabilities *CMR* experiences with optically thick coarse mesh cells.

In addition to the study of algorithms and applications pertaining to *NDA*, researchers have also studied fundamental modifications to the method itself. *NDA* adds a correction term to Fick's Law as shown in Eq. (2.41), guaranteeing closure of the high order and low order problems upon convergence of the solution. The exact form of this correction factor, however, may be changed, provided that the resulting modified Fick's Law will be used in the same manner as previously shown (i.e. it must be used to calculate the modified diffusion coefficients using the high order solution, and then to provide a relationship between the cell-edge current and cell-averaged scalar flux in the low order problem). In this regard, *NDA* is more of a method class than an individual method, as this correction factor can take many forms and the iterative performance can vary greatly depending on its prescription. We introduce equations for one of these special *NDA*-class methods, *partial-current Nonlinear Diffusion Acceleration (pNDA)* [35], as it is used in our work as a point of comparison in problems where *NDA* is observed to be unstable.

*pNDA* differs from traditional *NDA* in that it breaks net current into positive and negative components, thus calculating two modified coefficients for each face. We adjust the equations from [35] to make them applicable in 2-D, fixed source, monoenergetic, steady-state problems with isotropic scattering. The modified Fick's Law on which *pNDA* is based is,

$$J_{x,k+\frac{1}{2},j}^- = -\frac{D_{k+\frac{1}{2},j}(\phi_{k+1,j} - \phi_{k,j})}{2\Delta x_{k+\frac{1}{2}}} - \frac{1}{2} \tilde{D}_{k+\frac{1}{2},j}^- \phi_{k+1,j}, \quad (2.53)$$

$$J_{x,k+\frac{1}{2},j}^+ = -\frac{D_{k+\frac{1}{2},j}(\phi_{k+1,j} - \phi_{k,j})}{2\Delta x_{k+\frac{1}{2}}} + \frac{1}{2} \tilde{D}_{k+\frac{1}{2},j}^+ \phi_{k,j}, \quad (2.54)$$

$$J_{y,k,j+\frac{1}{2}}^- = -\frac{D_{k,j+\frac{1}{2}}(\phi_{k,j+1} - \phi_{k,j})}{2\Delta y_{j+\frac{1}{2}}} - \frac{1}{2} \tilde{D}_{k,j+\frac{1}{2}}^- \phi_{k,j+1}, \quad \text{and} \quad (2.55)$$

$$J_{y,k,j+\frac{1}{2}}^+ = -\frac{D_{k,j+\frac{1}{2}}(\phi_{k,j+1} - \phi_{k,j})}{2\Delta y_{j+\frac{1}{2}}} + \frac{1}{2} \tilde{D}_{k,j+\frac{1}{2}}^+ \phi_{k,j}. \quad (2.56)$$

Upon completion of an *SI* iteration, the modified diffusion coefficients are calculated in accordance with the modified Fick's Law using,

$$\tilde{D}_{k+\frac{1}{2},j}^{-(s+\frac{1}{2})} = -\frac{2}{\phi_{k+1,j}^{(s+\frac{1}{2})}} \left( J_{x,k+\frac{1}{2},j}^{-(s+\frac{1}{2})} + \frac{D_{k+\frac{1}{2},j} \left( \phi_{k+1,j}^{(s+\frac{1}{2})} - \phi_{k,j}^{(s+\frac{1}{2})} \right)}{2\Delta x_{k+\frac{1}{2}}} \right), \quad (2.57)$$

$$\tilde{D}_{k+\frac{1}{2},j}^{+(s+\frac{1}{2})} = \frac{2}{\phi_{k,j}^{(s+\frac{1}{2})}} \left( J_{x,k+\frac{1}{2},j}^{+(s+\frac{1}{2})} + \frac{D_{k+\frac{1}{2},j} \left( \phi_{k+1,j}^{(s+\frac{1}{2})} - \phi_{k,j}^{(s+\frac{1}{2})} \right)}{2\Delta x_{k+\frac{1}{2}}} \right), \quad (2.58)$$

$$\tilde{D}_{k,j+\frac{1}{2}}^{-(s+\frac{1}{2})} = -\frac{2}{\phi_{k,j+1}^{(s+\frac{1}{2})}} \left( J_{y,k,j+\frac{1}{2}}^{-(s+\frac{1}{2})} + \frac{D_{k,j+\frac{1}{2}} \left( \phi_{k,j+1}^{(s+\frac{1}{2})} - \phi_{k,j}^{(s+\frac{1}{2})} \right)}{2\Delta y_{j+\frac{1}{2}}} \right), \quad \text{and} \quad (2.59)$$

$$\tilde{D}_{k,j+\frac{1}{2}}^{+(s+\frac{1}{2})} = \frac{2}{\phi_{k,j}^{(s+\frac{1}{2})}} \left( J_{y,k,j+\frac{1}{2}}^{+(s+\frac{1}{2})} + \frac{D_{k,j+\frac{1}{2}} \left( \phi_{k,j+1}^{(s+\frac{1}{2})} - \phi_{k,j}^{(s+\frac{1}{2})} \right)}{2\Delta y_{j+\frac{1}{2}}} \right). \quad (2.60)$$

The partial currents in these equations are obtained using the numerical integration,

$$J_{x,k+\frac{1}{2},j}^{\pm(s+\frac{1}{2})} = \sum_{\Omega_{x,m} \geq 0} \Omega_{x,m} w_m \psi_{m,k+\frac{1}{2},j}^{(s+\frac{1}{2})} \quad \text{and} \quad J_{y,k,j+\frac{1}{2}}^{\pm(s+\frac{1}{2})} = \sum_{\Omega_{y,m} \geq 0} \Omega_{y,m} w_m \psi_{m,k,j+\frac{1}{2}}^{(s+\frac{1}{2})}. \quad (2.61)$$

With the modified diffusion coefficients calculated from the high order solution,  $pNDA$ 's modified Fick's Law then prescribes the current terms for the low order problem as,

$$J_{x,k+\frac{1}{2},j}^{-(s+1)} = -\frac{D_{k+\frac{1}{2},j}(\phi_{k+1,j}^{(s+1)} - \phi_{k,j}^{(s+1)})}{2\Delta x_{k+\frac{1}{2}}} - \frac{1}{2} \tilde{D}_{k+\frac{1}{2},j}^{-(s+\frac{1}{2})} \phi_{k+1,j}^{(s+1)}, \quad (2.62)$$

$$J_{x,k+\frac{1}{2},j}^{+(s+1)} = -\frac{D_{k+\frac{1}{2},j}(\phi_{k+1,j}^{(s+1)} - \phi_{k,j}^{(s+1)})}{2\Delta x_{k+\frac{1}{2}}} + \frac{1}{2} \tilde{D}_{k+\frac{1}{2},j}^{+(s+\frac{1}{2})} \phi_{k,j}^{(s+1)}, \quad (2.63)$$

$$J_{y,k,j+\frac{1}{2}}^{-(s+1)} = -\frac{D_{k,j+\frac{1}{2}}(\phi_{k,j+1}^{(s+1)} - \phi_{k,j}^{(s+1)})}{2\Delta y_{j+\frac{1}{2}}} - \frac{1}{2} \tilde{D}_{k,j+\frac{1}{2}}^{-(s+\frac{1}{2})} \phi_{k,j+1}^{(s+1)}, \text{ and} \quad (2.64)$$

$$J_{y,k,j+\frac{1}{2}}^{+(s+1)} = -\frac{D_{k,j+\frac{1}{2}}(\phi_{k,j+1}^{(s+1)} - \phi_{k,j}^{(s+1)})}{2\Delta y_{j+\frac{1}{2}}} + \frac{1}{2} \tilde{D}_{k,j+\frac{1}{2}}^{+(s+\frac{1}{2})} \phi_{k,j}^{(s+1)}. \quad (2.65)$$

The resulting iterative process for  $pNDA$  is identical to that of  $NDA$ , but with slightly altered equations. Upon completion of the  $SI$  mesh sweeps, the obtained flux and current distributions are used to calculate modified diffusion coefficients using Eqs. (2.57) - (2.60). These coefficients are then used to solve Eqs. (2.50) and (2.62) - (2.65) for the updated scalar flux distribution. This iterative process is then repeated until convergence is achieved.

### **2.1.6: *Transport Synthetic Acceleration (TSA)***

The final acceleration method for the  $SI$  iterative method we discuss in our review of the literature is *Transport Synthetic Acceleration (TSA)*. *Ramone, Adams, and, Nowak* [36] discuss  $TSA$  as an acceleration method which uses a low order transport problem for preconditioning. This low order problem is designed to converge a transport problem identical in configuration to the high order problem, except with a reduced scattering ratio. Since the spectral radius of  $SI$  is known

to be equal in the infinite homogeneous medium to the scattering ratio, the reduction of the scattering ratio results in a low order problem that is more rapidly convergent with  $SI$  than the high order problem.

After the transport sweep, the  $TSA$  low order problem takes the following form in two-dimensional Cartesian problem domains.

$$\begin{aligned} \left[ \mu_m \frac{\partial}{\partial x} + \eta_m \frac{\partial}{\partial y} + (1 - \beta c) \right] f_m^{(s+\frac{1}{2})}(x, y) \\ = \frac{(1 - \beta)c}{2\pi} F^{(s+\frac{1}{2})}(x, y) + \frac{c}{2\pi} \left[ \phi^{(s+\frac{1}{2})}(x, y) - \phi^{(s)}(x, y) \right] \end{aligned} \quad (2.66)$$

In this equation,  $\mu_m$  and  $\eta_m$  are the  $x/y$  direction cosines of angle  $m$ ,  $\beta \in [0,1]$  is a user specified input parameter which designates the low order problem's reduced scattering ratio, and  $s$  is the iteration index. When supplemented by the numerical integration,

$$F^{(s+\frac{1}{2})}(x, y) = \sum_{m=1}^M w_m f_m^{(s+\frac{1}{2})}(x, y) \quad (2.67)$$

it is possible to solve iteratively for  $f_m^{(s+\frac{1}{2})}(x, y)$  using the same mesh sweep algorithm as is used for the higher order problem. Upon convergence of the low order problem, the following equation is used to update the scalar flux iterate.

$$\phi^{(s+1)}(x, y) = \phi^{(s+\frac{1}{2})}(x, y) + F^{(s+\frac{1}{2})}(x, y) \quad (2.68)$$

While  $TSA$  is observed to be able to converge problems in far fewer iterations than the unaccelerated method, the cost of the low order calculation is of far more concern than with previously discussed acceleration methods. As opposed to solving a sparse matrix equation, mostly diffusion-like, the  $TSA$  method requires the solution of a full transport problem using  $SI$ , albeit a faster converging problem. Since any scattering ratio less than or equal to the problem's actual scattering ratio is usable for the  $TSA$  method, choice of this reduced scattering ratio is of immense

importance. While a smaller scattering ratio will obviously make the low order solution converge quicker, it will also provide less acceleration due to the larger discrepancy between the low and high order operators, ultimately requiring more iterations to achieve convergence. In many cases, a scattering ratio in the low order problem that is reduced too much was found to cause instability of the overall iterative method. On the other hand, while a scattering ratio of the low order problem close to that of the high order problem will cause the overall solution to converge in fewer iterations, these iterations will be costlier, as the low order problem will take longer to solve.

These issues can be illustrated through examples in the two extremes. In the case where the low order problem's reduced scattering ratio is equal to zero, the low order problem will converge in one mesh sweep over all angles. However, if the high order problem's scattering ratio was high, the low order problem's solution will be so far from the high order problem's that the acceleration will be minimal, or potential destabilizing. In the other extreme, if the low order problem's scattering ratio is equal to that of the high order problem, then *TSA* will converge in one iteration, as the low order problem is equivalent to converging the high order problem. However, this one iteration will have the same computational cost as solving the problem without acceleration. Given this important property of *TSA*, *Ramone*, *Adams*, and *Nowak* provide guidelines for selecting a reduced scattering ratio such that the method remains stable and provides significant acceleration. Additionally, these authors investigate the use of a reduced quadrature order for the low order problem, as a lower order quadrature set will result in a cheaper solution of the low order problem, at the presumed cost of an increased spectral radius of the combined method. While this was indeed found to be the case, it was shown that the spectral radius increased only slightly for a moderate reduction in quadrature order, thus yielding improved overall execution time. [36]

## **2.2: Parallel Transport Computations on Cartesian and Unstructured Grids**

The growing number of processors on modern computer architectures has prompted a new challenge for transport simulations, namely parallel iterative methods which effectively use the large number of available processors. Specifically, in this work we investigate methods for solving transport problems that can be effectively utilized in the massively parallel regime (100,000+

processors) and that can be extended to problems with unstructured tetrahedral spatial meshes. The motivation for using unstructured grids is obvious, as non-Cartesian geometries can be modeled more accurately with fewer cells. Implementation of any solution technique on unstructured grids, however, is more complicated than on Cartesian grids, with the extension being far simpler for some approaches than others. In this section, we discuss multiple approaches for parallelization of transport calculations, the amount of parallelism they can achieve, and their adequacy for extension to unstructured grids. In general, these parallelization techniques utilize domain decompositions, dividing one or more of the discretized independent variable spaces (space, angle, and energy) into sub-domains, along which the computational load is divided among processors.

### **2.2.1: Angular Domain Decomposition**

Decomposition of the angular domain's discrete ordinates for parallel execution of  $SI$  is mathematically simple in Cartesian geometry. Since angles are uncoupled over the course of an iteration, the mesh sweeps over different ordinates may be performed in parallel with no additional algorithmic changes. [37] Parallelism along the angular domain can achieve mid-range parallelism, 10s to 100s of processors, depending on the quadrature order used for a given application. [38] The solution algorithm for  $SI$  parallelized along the angular domain is therefore relatively unchanged from the serial algorithm, with sweeps for different angles conducted in parallel due to the lagged scattering source. The loop over angles is executed in parallel, with calculations such as the weighted summation of angular fluxes to obtain the scalar flux requiring either communication or shared memory.

With parallelization of  $SI$  along the angular domain not requiring significant alteration of the  $SI$  solution algorithm, it is readily extendible to multiple mesh types, including unstructured. For example, the *Method of Characteristics (MOC)*, a ray-tracing method, frequently uses meshes that are non-Cartesian, but not fully unstructured, meshing repeated structures such as nuclear fuel cells with mesh elements that model the exact problem geometry, but in a structured manner. *Boyd* [39] studied angularly parallel solution of  $MOC$  problems using a shared memory, OpenMP implementation. This implementation achieved 100% parallel efficiency on up to 12 processors, decreasing to 90% when cores were hyperthreaded. These results indicate the strong applicability of angularly parallel computations for mid-range parallel transport solution, with the independence

of parallel calculations and potential for shared memory parallelism providing excellent parallel efficiency.

Angularly parallel solution has also been studied on fully unstructured tetrahedral grids by *Yessayan* [38]. This implementation was completed in the THOR code [40, 41], which is also the code to which we add *PBJ-ITMM* and *IPBJ* capabilities in this work. *Yessayan*'s angularly parallel solution was implemented using MPI communication rather than the shared memory OpenMP implementation of the previous work, with the motivation of applicability to mid-range parallelism on high performance computing (HPC) platforms using 100s of processors. Testing this implementation on a distributed memory HPC demonstrated similar parallel efficiencies to those reported by *Boyd* [39], but decreasing slightly as the processor count extends into the 100s.

### **2.2.2: Energy Domain Decomposition**

Parallelization along the energy domain occurs at the outer iteration level, as the outer iterations iterate over energy groups. Energy groups are decoupled from each other, hence the inner iterative solutions are performed in parallel across groups. While this is conceptually similar to the angular domain decomposition discussed in the previous section, two key differences result in parallelism along the energy domain being widely regarded as less desirable to parallelism along the angular domain for transport solution. Firstly, the number of energy groups is very often smaller than the number of discrete angles, reducing the number of processors that can be utilized, and therefore the maximum potential parallel speedup. [38]

Additionally, decomposing the energy domain can require imposing additional lagged terms that are not required to be lagged in the serial *SI* solution algorithm. While all angles are decoupled over the course of an inner iteration, all energy groups are not necessarily decoupled over the course of an outer iteration. It is common practice for outer iterations to loop over energy groups (fastest to slowest), using a current-iterate or lagged between-group source term for faster or slower groups respectively, rather than lagging this term for all energy groups. This practice allows down-scattering to be resolved during a single outer iteration, only requiring up-scattering and fission production to be resolved iteratively. The required number of iterations can consequently increase as a result of parallelization along the energy domain. In problems that do not have up-scattering or fission, in fact, outer iterations are not typically used, as there is no mutual

coupling between energy groups, making parallelization along the energy domain poorly suited for such applications. [37]

While the energy domain decomposition is not frequently used, its extension to unstructured grids is as conceptually simple as with the angular domain decomposition, requiring no fundamental modification to the used algorithm to obtain a given iterate's solution.

### 2.2.3: *Koch-Baker-Alcouffe (KBA) Spatial Domain Decomposition on Cartesian Grids*

The angular domain decomposition is the most common technique for mid-range parallelism of the neutron transport solution. This technique is limited in scale, however, by the size of a problem's angular quadrature, which typically does not exceed 100s of angles. To scale past this range of parallelization into processor counts of 1000s, 10,000s, and ultimately 100,000s+ (massively parallel regime), parallelization must be applied along the spatial domain, as this is the only domain with enough discrete elements to support processor counts of this scale.

The *Koch-Baker-Alcouffe (KBA)* [42, 43] spatial domain decomposition splits the discretized spatial domain in such a manner as to allow for spatially parallel solution of the *SI* method. *KBA* is currently used for massively parallel execution of the *SI* iterative method in production-level transport codes, such as PARTISN [44], that utilize structured meshes. The *KBA* decomposition takes advantage of the fact that, in multidimensional problems, the required information (all incoming angular fluxes) is usually known for multiple cells simultaneously. Parallelization is achieved along the spatial domain through decomposition into sub-domains, as depicted for a 2-D Cartesian mesh for solution on four processors in Fig. 2.1.

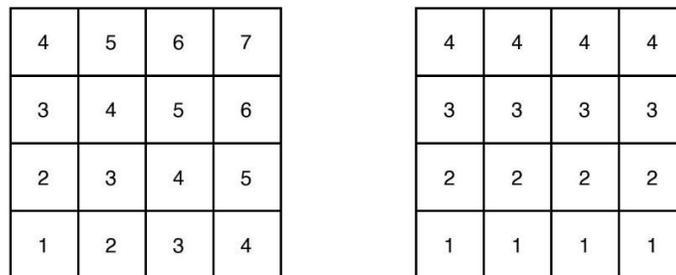


Figure 2.1: Sub-domains for a 4-processor *KBA* decomposition of 2-D Cartesian mesh. Left image indicates the order of solution for an angle in the first quadrant, right, the processors to which sub-domains are assigned

The left image in this figure indicates the order in which the sub-domains are solved for an angle in the first quadrant, with sub-domains containing the same number being solved simultaneously on different processors. The solution of a sub-domain consists of a local, serial mesh sweep over all cells within the sub-domain for a specified angle. The parallel sweep over sub-domains progresses as a wavefront across the domain, reaching full parallelism along the diagonal (4<sup>th</sup> stage in this example). The image on the right-hand-side of Fig. 2.1 displays the processor to which the solution of a given sub-domain is assigned. Considering this assignment of sub-domains to processors as well as the wavefront progression on the left, after the fully parallelized solution of the diagonal sub-domains, at each subsequent stage of the wavefront progression, processors become idle at the same stage for which they would be required in the parallel sweep of another discrete ordinate. To reduce processor downtime, angular pipelining is used, beginning the sweep of a new angle after the previous angle's sweep finishes solution of the diagonal sub-domains. Pipelining is also possible along energy groups.

The synchronicity of these spatial sub-domains is imperative, as flux continuity must be applied on sub-domain interfaces to conduct the synchronous mesh sweep required by the *SI* method. Since the *KBA* decomposition only dictates the grouping of cells it will not affect the converged solution beyond arithmetic precision. Additionally, as the decomposition is synchronous, it will not alter the number of iterations required for convergence as the number of participating processors changes.

While the use of multiple processors with no penalty to the number of required iterations reduces the execution time of transport calculations, execution time is not simply scaled by the number of processors used, as the addition of processors has an inherent penalty of increased processor downtime and communication costs. *Fischer* and *Azmy* [43, 45] study the computational efficiency of parallel transport algorithms for the *SI* method, comparing the *KBA* decomposition to the angular domain decomposition. Quantitatively, the computation time associated with the local solution of a single sub-domain in a 3-D transport problem solved with the *KBA* decomposition is, [43]

$$T_{cpu} = \left( \frac{N_x}{P_x} \frac{N_y}{P_y} \frac{N_z}{K_b} \frac{N_a}{A_b} \right) \frac{N_{flops}}{R_{flops}} \quad (2.69)$$

where  $N_u$  is the number of cells in the  $u^{th}$  spatial dimension,  $K_b$  is the number of concurrent divisions of the  $z$  direction (blocks which are calculated concurrently on a single processor),  $A_b$  is the number of such concurrent blocks in the angular domain,  $N_a$  is the number of angles used,  $N_{flops}$  is the average number of required floating-point operations for a single cell's solution in a single direction, and  $R_{flops}$  is the average rate at which the processor is able to perform said operations. The communication time associated with a local sub-domain's solution is then,

$$T_{msg} = t_0 + \frac{N_{msg}}{B}, \quad (2.70)$$

where  $t_0$  is the communication latency,  $B$  is the bandwidth, and  $N_{msg}$  is the size of messages that must be exchanged. The total computation wall-clock time is then given by,

$$T_{comp} = [(P_x + P_y - 1) + (N_{sweep} - 1)]T_{cpu}. \quad (2.71)$$

where  $N_{sweep}$  is the number of mesh sweeps. Additionally, the total communication time is given by,

$$T_{comm} = [2(P_x + P_y - 2) + 4(N_{sweep} - 1)]T_{msg}. \quad (2.72)$$

These equations approximate the computation and communication times associated with the *KBA* decomposition, taking into account both the cost of passing messages between processors and the processor idle time associated with *KBA*.

In computational testing, problems solved with the *KBA* decomposition spent a fraction of their execution time communicating that is similar to problems solved with the angular domain decomposition. The *KBA* decomposition also has an advantage over the angular domain decomposition in that it can be used for massively parallel transport solutions, as the number of processors in these types of problems typically far surpasses the number of angles used.

## 2.2.4: Parallel Mesh Sweeping on Unstructured Grids

The previous comments regarding the *KBA* spatial domain decomposition for parallel execution of the *SI* iterative method have all assumed the spatial mesh to be structured. Considering Fig. 2.1, it is clear that straightforward scheduling of parallel sub-domain solutions is an attractive feature of *KBA* that does not extend from Cartesian to unstructured grids. Scheduling parallel sub-domain solutions on unstructured grids requires complex algorithms, previous work on which is summarized in this section.

The main difficulty identified for mesh sweeps on an unstructured grid, parallel or serial, is determining the scheduling precedence of calculations that retains synchronicity. Even in a serial sweep, the scheduling is more complex than in structured grids, where the scheduling of serial cell sweeps is determined trivially from the cell indices and the angular octant in which the swept angle resides. Firstly, the sweep order in unstructured grids varies between individual angles, i.e. discrete ordinates, rather than just by angular octant. Additionally, it is possible for cells within the mesh to be mutually dependent on each other. The cyclic dependencies of these cells are typically alleviated by breaking the dependence of a single selected cell on its neighbor in the cycle. This can be accomplished by simply lagging by one iteration, the cell-edge angular flux for one of the cells involved in the mutual dependency. This process can increase the number of required iterations but does not alter the solution to which the iterations converge, to within the convergence criterion. [46]

Despite these added complexities, *Pautz* [46] notes that the scheduling theory required for determining the sweep order of spatial cells for the serial calculation is relatively simple. For parallel execution, however, the scheduling becomes far more problematic. The scheduling of such a parallel sweep in unstructured grids is cast as a graph problem which is NP-complete. *Pautz* explains that this indicates it is not possible for an algorithm to efficiently determine the optimal schedule except in special cases. Practically, heuristics must be used to determine the scheduling of cell-angle sweeps in the problem, which will not produce the optimal scheduling. [46, 47] While *Pautz* presents multiple heuristic algorithms for parallel mesh sweeps, all have the shortcoming that they have no provable worst-case-scenario scaling of required execution time to construct the sweep schedule with respect to the mesh size. Further work by *Kumar* [48] develops an algorithm

for performing this scheduling task that provably scales in the worst-case-scenario almost linearly with respect to mesh size.

Complexities notwithstanding, research has been performed on the execution of parallel mesh sweeps on unstructured grids, much of which focuses on decreasing the processor idle time that results from the sub-optimal parallel sweep paths that emerge from the scheduler. Work presented by *Plimpton* [47] tests two enhancements on a parallel sweep algorithm, designed to increase the parallel efficiency by decreasing idle processor time. The first of these enhancements modifies the spatial domain decomposition rather than the sweep algorithm itself. This enhancement, termed *KBA Priority*, attempts to replicate the *KBA* decomposition of a Cartesian grid as closely as possible in unstructured grids. This approach decomposes the unstructured grid into sections that are approximately columnar, then decomposes these approximate columns into layers as uniformly as possible. The result of the *KBA Priority* is a spatially decomposed unstructured grid with a sub-domain structure that is similar to that of a Cartesian grid, thus improving the ability to sweep across it in parallel while minimizing processor idle time.

The second enhancement presented by *Plimpton*, termed *3-D Priority*, modifies the order in which cells are solved by a given processor. For a processor, at any stage, there will likely be multiple cells assigned to that processor for which the information is available to solve. The *3-D Priority* approach decreases processor idle time by prioritizing the solution of cells that are of greater importance to other processors. This determination is made by assessing which cell is most upstream. The dot product of a cell's centroid and the discrete ordinate that is being swept determines which cell is most upstream, with a lower evaluated dot product corresponding to being further upstream.

Numerical testing on up to 256 processors demonstrates significantly improved parallel efficiency of the parallel sweep algorithm with both of these enhancements, *KBA Priority* achieving greater performance gains than *3-D Priority*. Since these tests were conducted only on up to 256 processors, communication costs were minimal, with parallel efficiency mainly a function of processor idle time.

Another approach for decreasing processor idle time is mesh decomposition overloading, presented by *Pautz* and *Bailey*. [49] Mesh decomposition overloading is the practice of increasing the number of sub-domains in a spatial domain decomposition past the ideal amount for the number of available processors, based on a user prescribed input. For example, considering the 2-D

Cartesian example in Fig. 2.1, an  $8 \times 8$  grid (or any other grid larger than  $4 \times 4$ ) of sub-domains may be imposed on the mesh for the same 4-processor solution. Doing so compensates for the fact that, in unstructured grids, the parallel wavefront does not move across the domain in an easily predictable manner. With each processor responsible for the solution of a greater number of smaller sub-domains, the probability at any given moment of a processor possessing a sub-domain, whose required inputs for solution are known, increases. Consequently, the increased idle time associated with the aforementioned unpredictable wavefront progression is mitigated. Weak scaling numerical testing of this approach demonstrated overloading to significantly decrease the observed execution times. This weak scaling study was conducted on up to 10,000s of processors.

One final approach we review, presented by *Adams* [50], is to require a mesh to be Cartesian on a coarse level, while allowing it to be arbitrarily unstructured on a fine level. The coarse Cartesian mesh supports the *KBA* decomposition, the structured configuration of which avoids the parallel scheduling complications associated with unstructured grids, allowing the optimal structured grid *KBA* scheduling to be used. The local sub-domain solutions then serially sweep over the fine level unstructured mesh. This approach was demonstrated to have computational efficiency equivalent to *KBA* in Cartesian grids, while allowing sub-domains to be locally unstructured. This approach is effective for geometries with repeating, coarsely Cartesian regions, such as nuclear fuel assemblies, which were the geometries tested in [50].

## **2.2.5: *Parallel Block Jacobi (PBJ) Spatial Domain Decomposition***

*Parallel Block Jacobi (PBJ)* is another spatial domain decomposition that can be used for spatially parallel transport calculations. Effective methods and algorithms for transport calculations using the *PBJ* decomposition is the main topic of this work. Our motivation for using this decomposition is that, by virtue of it being asynchronous, it does not have the scheduling difficulties incurred by the *KBA* decomposition in unstructured grids. As with *KBA*, *PBJ* divides the spatial mesh into sub-domains, each of which may contain one or multiple cells. The difference between the two decompositions is that *PBJ* is asynchronous, meaning that over the course of an iteration, the sub-domains are unable to communicate with each other. While this alleviates the scheduling difficulty, it also means that the standard *SI* iterative method cannot be used with this decomposition. This follows from the fact that the incoming angular fluxes to sub-domains must

be lagged by an iteration precipitating the asynchronous nature of the *PBJ* decomposition, rendering global synchronous mesh sweeps impossible. In the subsequent sections, we review iterative methods for transport calculations using the *PBJ* spatial domain decomposition. Ultimately, we are also interested in *Parallel Gauss-Seidel (PGS)* methods, but since these are effectively a slight modification of *PBJ* they will not be covered in further detail here.

## 2.2.6: Iterative Methods for the *PBJ* Spatial Domain Decomposition

*Yavuz* and *Larsen* [51] developed a method for implementing *SI* in asynchronous spatial domain decompositions which fundamentally alters the *SI* iterative sequence. This method was later formally termed *Inexact Parallel Block Jacobi*, which is reviewed below.

*Inexact Parallel Block Jacobi (IPBJ)* is an iterative method that is compatible with the *PBJ* decomposition, depicted in Fig. 1.1. Formally, *IPBJ* is described by the same set of mathematical equations as *SI*, but with the incoming angular flux term split so that it is lagged if the incoming angular flux is on a sub-domain interface, and is otherwise not lagged. *IPBJ* is executed by running a single local mesh sweep in each sub-domain over each direction, inferring the incoming angular fluxes at sub-domain boundaries from either the outgoing angular fluxes from adjacent upstream sub-domains computed in the previous iteration, or from the boundary conditions. Since *IPBJ* uses mesh sweeps to obtain local solutions, the execution time and memory burden associated with a sub-domain's local solution scales linearly with the number of cells in the sub-domain.

Another iterative strategy used with the *PBJ* decomposition is the *Parallel Block Jacobi – Integral Transport Matrix Method (PBJ-ITMM)*. [52] *PBJ-ITMM* lags only the quantities required to be lagged due to the asynchronicity of the *PBJ* decomposition; the incoming angular fluxes at sub-domain interfaces. It is equivalent to the *IPBJ* method without lagging the scattering source. The *PBJ-ITMM* method is therefore executed by obtaining a fully-collided local transport solution in each sub-domain assuming the incoming angular fluxes at the sub-domain boundaries to be known.

The local sub-domain solution for *PBJ-ITMM*, requires the construction of the associated matrices prior to the beginning of iterations. These matrices are comprised of the coupling factors between cells, essentially comprising the sub-domain's response matrices. The traditional algorithm for constructing these matrices is the differential mesh sweep [52, 53], which we briefly

describe here, but do not mathematically formulate until Sec. 3.2.3, as we use it in the development of a new algorithm. The differential mesh sweep is a nested sweep in that a sweep is initiated from each cell through all downstream cells. This sweep solves for the derivative of a quantity in one cell with respect to the sub-domain boundary angular fluxes and the cell-averaged scalar fluxes in upstream cells, which is the coupling (or response) of one cell's flux quantities to a perturbation in boundary or cell-averaged fluxes in another cell. In this regard, *ITMM* can be thought of as a response matrix method. [54] This differential mesh sweep is used to determine, for all cells, the coupling of the cell-averaged scalar flux to the same quantity in all other cells as well as the sub-domain boundary incoming angular fluxes. Additionally, the differential mesh sweep determines the coupling of the sub-domain boundary's outgoing angular fluxes to all cell-averaged scalar fluxes and sub-domain boundary incoming angular fluxes. The four resulting matrices are used each iteration to first obtain the cell-averaged scalar flux distribution over the sub-domain from the sub-domain boundary incoming angular fluxes, then to obtain the sub-domain boundary outgoing angular fluxes from both the sub-domain boundary incoming angular fluxes and the just-computed cell-averaged scalar fluxes. [52]

The use of matrix operations causes a nonlinear scaling of computational cost and memory burden for the local solution with respect to sub-domain size. *Powell* demonstrates the magnitude of the memory constraint for given sub-domain sizes, showing that a 3-D problem with a modest quadrature order can exceed 30GB of required RAM using sub-domains with as few as  $12 \times 12 \times 12$  cells. [55]

To mathematically demonstrate the difference between *SI*, *IPBJ*, and *PBJ-ITMM*, we provide the discretized balance equation for these three methods respectively as follows. [4] These equations are for a two-dimensional, steady state problem with no fission, isotropic scattering, and an isotropic external source. Additionally, we consider the case for the *PBJ* methods where sub-domains contain a single cell. We present the equations for *SI*, *IPBJ*, and *PBJ-ITMM*, respectively, for this sample problem as it will be a model problem used in the spectral analyses described later in this work.

$$\begin{aligned} v_{x,m,k,j} \left( \psi_{m,k,j}^{x+, (s+1)} - \psi_{m,k,j}^{x-, (s+1)} \right) + v_{y,m,k,j} \left( \psi_{m,k,j}^{y+, (s+1)} - \psi_{m,k,j}^{y-, (s+1)} \right) + \psi_{m,k,j}^{(s+1)} \\ = \frac{c_{k,j}}{4\pi} \phi_{k,j}^{(s)} + \frac{1}{4\pi \Sigma_{t,k,j}} q_{k,j} \end{aligned} \quad (2.73)$$

$$\begin{aligned}
v_{x,m,k,j} \left( \psi_{m,k,j}^{x+, (s+1)} - \psi_{m,k,j}^{x-, (s)} \right) + v_{y,m,k,j} \left( \psi_{m,k,j}^{y+, (s+1)} - \psi_{m,k,j}^{y-, (s)} \right) + \psi_{m,k,j}^{(s+1)} \\
= \frac{c_{k,j}}{4\pi} \phi_{k,j}^{(s)} + \frac{1}{4\pi \Sigma_{t,k,j}} q_{k,j}
\end{aligned} \tag{2.74}$$

$$\begin{aligned}
v_{x,m,k,j} \left( \psi_{m,k,j}^{x+, (s+1)} - \psi_{m,k,j}^{x-, (s)} \right) + v_{y,m,k,j} \left( \psi_{m,k,j}^{y+, (s+1)} - \psi_{m,k,j}^{y-, (s)} \right) + \psi_{m,k,j}^{(s+1)} \\
= \frac{c_{k,j}}{4\pi} \phi_{k,j}^{(s+1)} + \frac{1}{4\pi \Sigma_{t,k,j}} q_{k,j}
\end{aligned} \tag{2.75}$$

where we define,

$$v_{u,m,k,j} \equiv \frac{|\Omega_{u,m}|}{\Sigma_{t,k,j} \Delta u_{k,j}}, \quad u = x \text{ or } y. \tag{2.76}$$

In these equations,  $s$  is the iterations index,  $k$  and  $j$  refer to the spatial cell  $(k, j)$ , and  $m$  is the discrete angle index. Also, the superscript  $u\pm$  refers to the outgoing/incoming quantity on the  $u$  edge of the cell where  $u = x$  or  $y$ . Note that for Eqs. (2.74) and (2.75), the incoming angular flux terms are only universally lagged due to the single cell sub-domains. In implementation, multiple cells may be used in each sub-domain, in which case, only edges on a sub-domain boundary have their incoming angular flux terms lagged.

We see from these three equations that *SI*, *IPBJ*, and *PBJ-ITMM* differ from each other based on the terms they lag by an iteration. From Eq. (2.74), it is easy to notice that the local *IPBJ* solution of a sub-domain over a single iteration is equivalent to one local *SI* iteration over that sub-domain with the lagged incoming fluxes used as boundary conditions. This results in the same triangular structure of the matrix associated with *SI* that allows iterative solution via mesh sweep. This is what mathematically affords *IPBJ* the advantage of being able to utilize the computationally effective mesh sweeps as well. However, this equation also demonstrates a shortcoming which will be described in the following section; that *IPBJ* has the worst rate of convergence of these three methods. This is intuitive from these equations, as *IPBJ* lags every term that is lagged in either of the other methods.

Analyzing Eq. (2.75), we see that *PBJ-ITMM* is the only method which does not lag the scalar flux within the scattering term. Consequently, a local sub-domain non-iterative solution with

*PBJ-ITMM* cannot use the mesh sweep algorithm for solution, as the associated matrix is not triangular.

*PBJ* methods are of interest in our work for their simpler applicability to parallel solution on unstructured grids. Without the scheduling difficulties associated with *SI* in this capacity, *IPBJ* is conceptually simple to extend to unstructured grids, with each sub-domain simply executing the traditional set serial of mesh sweeps over all angles. *PBJ-ITMM*'s iterative solution is a set of matrix/vector operations that are agnostic to the type of mesh used, making it also conceptually simple to extend to unstructured grids, once the *ITMM* matrices are constructed. Previously developed construction algorithms for these matrices exist, however, they are not readily extendible to unstructured grids. Consequently, development, implementation, and testing of an algorithm capable of performing this task is one of the main objectives of our work presented here.

### **2.2.7: Iterative Convergence of Methods Using *PBJ* Spatial Domain Decomposition**

Fourier analysis and numerical experiments have shown that both *PBJ* methods mentioned above lack robustness in two limits: (1) as the scattering ratio increases towards unity; and (2) as the optical thickness of cells in the problem decreases towards zero, with the largest magnitude eigenvalue occurring at the origin in Fourier space. In the infinite medium, both methods fail to converge for configurations with unit scattering ratio. [4] While this behavior is analogous to that of the *SI* method, the dependence of the spectral radius on cell optical thickness is not. This lack of robustness in optically thin cells is discussed and addressed in detail throughout this work.

While the lack of robustness with thinning cells and increasing scattering ratio is the main takeaway from studies on the iterative convergence of *PBJ* methods, the spectra of these methods differ depending on the spatial discretization method used. For *AHOT-N0*, the general trends described above apply. However, with the *DD* method, *Azmy*, *Anistratov*, and *Zerr* found *PBJ-ITMM* to be unconditionally unstable in the infinite medium, with the unstable eigenvalue being negative. [4] These authors also found the spectral radius of *IPBJ* to be larger than or equal to that of *PBJ-ITMM* for the same problem. This is an expected result, as *IPBJ* lags all computed quantities that *PBJ-ITMM* lags, plus the scattering term. Additionally, their study quantified the penalty from lagging the scattering source. From these studies of the iterative convergence of *IPBJ*

and *PBJ-ITMM* the primary challenges are identified to be achieving: (1) stability and robustness with increasing scattering ratio; (2) stability and robustness with decreasing cell thickness; and (3) stability when these iterative schemes are used with the *DD* method. [56]

### 2.2.8: Acceleration of *PBJ*-Type Methods

Recent work has been reported seeking to address the aforementioned challenges, effectively accelerating *PBJ* methods. *Rosa*, *Warsa*, and *Chang* studied the use of *TSA* as an acceleration method for *IPBJ*. When using *Traditional TSA (TTSA)*, the authors found that robustness could be achieved in optically thin cells, but only for significantly large scattering ratios used in the low order *TSA* problem. Additionally, the resulting method was found to be divergent for many smaller choices of reduced scattering ratio. This prompted the study of *Modified TSA (MTSA)* as an accelerator for *IPBJ*. *MTSA*, unlike *TTSA*, does not reduce the total cross section with the scattering cross section. *IPBJ* accelerated using *MTSA* was observed to be stable for all choices of reduced scattering ratio. This acceleration technique also achieved robustness in optically thin cells with scattering ratios smaller than one, and the resulting method was found to have convergence properties far less sensitive to cell optical size. [57]

To address the instability when used with the *DD* method, *Hoagland*, *Azmy*, and *Zerr* studied a modified *PBJ* method in which the solution to one modified iteration was defined as the arithmetic average of the solutions of two successive *PBJ-ITMM* iterations. This idea was founded on the realization that an eigenvalue equal to  $-1$  would cause the iterative solution to oscillate equidistantly around the limit for the corresponding Fourier mode. Fourier analysis of the modified iterative method showed the previously unstable mode to have an eigenvalue of zero. The higher frequency modes, however, remained unstable. This problem was addressed by executing *PBJ-ITMM* to a looser convergence criterion, then executing *SI* to the specified criterion, as *SI* rapidly converges the high frequency modes. [56]

While the instability of *PBJ*'s high frequency modes is a trait specific to the *DD* method, the technique of utilizing sweep-based iterative methods as a means of acceleration for *PBJ* methods is also used successfully to mitigate the slowdown in optically thin cells. This is one of the two main objectives of this work, with associated reference material available in [58 - 60].

### 2.2.9: Hybrid Domain Decomposition

The final technique used for parallel solution of the neutron transport we review is the hybrid domain decomposition. This approach decomposes and solves in parallel along multiple discrete independent variables, typically space and angle. This approach, presented by *Song* in [61], uses shared memory (OpenMP for CPU, CUDA for GPU) parallelism along the angular domain and distributed memory MPI parallelism along the spatial domain. An entire node of an HPC is then assigned to each *KBA* sub-domain rather than a single processor. The processors that comprise a node then solve a given sub-domain, sweeping the angles in parallel. The general concept of this approach is to reduce the number of *KBA* sub-domains required for execution on a given number of processors compared to when only the spatial domain is decomposed. This reduces the amount of off-node communication that is required to obtain the transport solution, which is vastly more time consuming than on-node communication or shared memory parallelism. Numerical testing in [61] demonstrates improved computational performance using this hybrid approach compared to MPI exclusively. Additionally, the combination of MPI and OpenMP is observed to result in slightly improved parallel efficiencies compared to MPI and CUDA. The extension of a hybrid domain decomposition to unstructured grids is the combination of the challenges associated with extending each individual decomposition utilized.

# **Chapter 3:**

## ***P-PI-SI & P-IP-SI Spectral Analysis & Formulation of the Green's Function ITMM Construction Algorithm***

### ***3.1 P-PI-SI & P-IP-SI Theoretical Analysis***

As previously stated, one of the ultimate objectives of this work is to provide robust *PBJ*-type iterative methods that are scalable to massively parallel solution on unstructured spatial meshes. To do so, we must address the iterative slowdown *PBJ* methods incur in optically thin cells. As seen in [4], the spectral radii of *PBJ-ITMM* and *IPBJ* tend towards unity as the optical thickness of the problem's spatial cells tends towards zero. We conjecture that this iterative slowdown is the result of the increased average number of cells traversed by particles before they are lost via absorption or leakage when the cells are optically thin. After each *PBJ* iteration, information is only exchanged between adjacent sub-domains. If we consider each iteration to be generating new information, namely change in the new flux iterate as it approaches the iterative limit, then it takes many iterations for that information to reach distant sub-domains. As the sub-domains become optically thin, the information generated in a sub-domain during an iteration becomes increasingly important for the convergence of the solution at distant sub-domains, resulting in an increase in the number of required iterations. This conjecture is supported by the previously noted fact that the exact transport matrix approaches a tridiagonal structure as cells become thick in a 1-D problem [21], which is interpreted as distant cells becoming decreasingly coupled to each other in this limit. We deduce from this fact that as cells become optically thin, distant cells become increasingly coupled to each other.

With *PBJ*'s ineffectiveness at converging the solution for problems involving long-distance streaming that we conjecture is due to its asynchronicity, we propose a new hybrid approach, pairing *PBJ*-type methods with traditional sweep-based methods. The motivation behind our new class of methods is that the synchronous nature of sweep-based methods (*SI* for the entirety of this chapter) will resolve the long-distance streaming in a problem, while *PBJ-ITMM*

will resolve the local scattering which would otherwise require many *SI* iterations to resolve. With *IPBJ* unable to resolve local collisions in a single iteration, it was predicted that *PBJ-ITMM* would prove the more effective *PBJ*-type method in this hybrid approach. For this reason, we include analysis and testing of hybrid formalisms with both *PBJ-ITMM* and *IPBJ* to confirm that *PBJ-ITMM* is necessary and that our understanding of the underlying mechanism of our hybrid approach is valid.

The analysis and testing for our hybrid methods is conducted on 2-D, Cartesian grids with serial execution during numerical tests. This simplified configuration is utilized at this stage as it significantly expedites the development process. Since our motivation of studying *PBJ* methods is to provide alternatives to the complex sweep algorithms associated with spatially parallel *SI* on unstructured grids, we must assume that the mesh sweeps used by our hybrid methods are not spatially parallelizable in our target application. The first and simplest hybrid method that we propose is *Parallel Block Jacobi with Source Iteration Preconditioning* (*P-PI-SI* or *P-IP-SI* when using *PBJ-ITMM* or *IPBJ* respectively), an iterative strategy that consists of a *PBJ-ITMM* or *IPBJ* iteration followed by an *SI* iteration. Other hybrid variants will be detailed later in this work, their development motivated by the performance observed with *P-PI-SI* and *P-IP-SI*. The preconditioning hybrid approach is used primarily as a proof of concept to demonstrate the viability of using a sweep-based method to mitigate the slowdown of *PBJ*-type methods in optically thin cells. Due to the global mesh sweeps and our constraint that these will not be spatially parallelized on unstructured grids, the preconditioning approach imposes a large penalty to the potential degree of parallelism of the *PBJ* method. The sequential execution of *PBJ* followed by *SI*, however, makes this combination of the two methods ideal for theoretical analysis to gain insight into the expected iterative performance of this hybrid approach and facilitate the development presented in the following chapter. Spectral analysis of *P-PI-SI* and *P-IP-SI* to obtain mathematical expressions for their iterative convergence rates is the subject of this section.

### **3.1.1 Formulation of *P-PI-SI* & *P-IP-SI***

Before we introduce our analysis of *P-PI-SI* and *P-IP-SI*, there are a few important considerations to note. Firstly, it must be noted that a *P-PI-SI* (or *P-IP-SI*) iteration consists of both a *PBJ-ITMM* (or *IPBJ*) and a single *SI* iteration, making a single combined-iteration's

computational cost equal to the sum of the computational costs of each stage. Also, we note that the *SI* method lags the cell-averaged scalar flux used to compute the scattering source, whereas *PBJ* methods lag the incoming angular fluxes at sub-domain interfaces (and the scattering source if *IPBJ* is used). This means that the *PBJ* step provides the cell-averaged scalar flux for the *SI* step and the *SI* step provides incoming angular fluxes for the *PBJ* step (plus the cell-averaged scalar flux if *IPBJ* is used).

For simplicity, we analyze the iterative performance of *P-PI-SI* and *P-IP-SI* in Cartesian, 2-D spatial meshes for steady-state, one energy group, non-multiplying problems with isotropic scattering under the assumption that general performance trends extend to other configurations. While *PBJ* methods can have sub-domains of varying size in number of cells, for the purpose of spectral analysis we consider the case where each sub-domain consists of only one spatial cell. We consider this case since single cell sub-domains will result in the slowest iterative convergence rate, i.e. the largest spectral radius, providing the worst-case scenario in terms of iterative convergence rate. For *P-PI-SI* the first step of the iterative process is a *PBJ-ITMM* iteration, mathematically expressed for the described problem type by,

$$\begin{aligned} v_{x,m,k,j} \left( \psi_{m,k,j}^{x+,(s+\frac{1}{2})} - \psi_{m,k,j}^{x-,(s)} \right) + v_{y,m,k,j} \left( \psi_{m,k,j}^{y+,(s+\frac{1}{2})} - \psi_{m,k,j}^{y-,(s)} \right) + \psi_{m,k,j}^{(s+\frac{1}{2})} \\ = \frac{c_{k,j}}{4\pi} \phi_{k,j}^{(s+\frac{1}{2})} + \frac{1}{4\pi \Sigma_{t,k,j}} q_{k,j} , \end{aligned} \quad (3.1)$$

$$\psi_{m,k,j}^{(s+\frac{1}{2})} = \frac{1 + \alpha_{u,m,k,j}}{2} \psi_{m,k,j}^{u+,(s+\frac{1}{2})} + \frac{1 - \alpha_{u,m,k,j}}{2} \psi_{m,k,j}^{u-,(s)} , \quad u = x \text{ or } y , \quad (3.2)$$

and

$$\phi_{k,j}^{(s+\frac{1}{2})} = \sum_{m=1}^M w_m \psi_{m,k,j}^{(s+\frac{1}{2})} , \quad (3.3)$$

where  $v_{u,m,k,j}$  was previously defined in Eq. (2.76).

Equations (3.1) - (3.3) represent the particle balance equation, *Weighted Diamond Difference (WDD)* auxiliary equation, and numerical integration of the angular flux into the scalar flux, respectively. Note that these equations are reiterated from the literature review, more specifically from [4], but with different iteration indices. In these equations the superscript ( $s$ ) indicates the iteration index, where any term with an  $s$  superscript is a quantity which is inferred from the previous iteration or initial guess, and any term with an  $\left(s + \frac{1}{2}\right)$  superscript is an unknown quantity which is solved for in the present *PBJ-ITMM* iteration. The subscripts  $k$  and  $j$  refer to the spatial cell  $(k, j)$  of a  $K \times J$  mesh,  $m$  refers to an individual discrete direction in a quadrature comprised of  $M$  angles, and  $w_m$  corresponds to that angle's angular weight. These three equations use an input of the lagged incoming angular fluxes to compute as output the cell-averaged scalar flux and outgoing angular fluxes. The cell-averaged scalar flux is the solution of the *PBJ-ITMM* iteration stage which is then input into the *SI* step's scattering source. Before formulating the equations for the *SI* step, we note that Eq. (3.2) is valid for any *WDD* scheme, where the weighting factors ( $\alpha$ ) take a value that corresponds to the selected numerical method. In this analysis, we consider the *AHOT-NO* method, which determines unique weighting factors for each cell / angle / dimension combination using the expression,

$$\alpha_{u,m,k,j} = \coth\left(\frac{\Sigma_{t,k,j}\Delta u_{k,j}}{2|\Omega_{u,m}|}\right) - \frac{2|\Omega_{u,m}|}{\Sigma_{t,k,j}\Delta u_{k,j}}, \quad (3.4)$$

where  $u$  is the dimension subscript  $u = x$  or  $y$ . In the *P-PI-SI* iterative method, after the *PBJ-ITMM* iteration yields the cell-averaged scalar flux, the next step is to use this quantity in the scattering source of an *SI* iteration to obtain the outgoing angular fluxes, as these angular fluxes will be used as the incoming angular fluxes for adjacent sub-domains in the ensuing *PBJ-ITMM* step. For the previously described 2-D problem type, the following equations mathematically describe the *SI* step of *P-PI-SI*.

$$\begin{aligned} v_{x,m,k,j} \left( \psi_{m,k,j}^{x+, (s+1)} - \psi_{m,k,j}^{x-, (s+1)} \right) + v_{y,m,k,j} \left( \psi_{m,k,j}^{y+, (s+1)} - \psi_{m,k,j}^{y-, (s+1)} \right) + \psi_{m,k,j}^{(s+1)} \\ = \frac{c_{k,j}}{4\pi} \phi_{k,j}^{(s+\frac{1}{2})} + \frac{1}{4\pi\Sigma_{t,k,j}} q_{k,j} \end{aligned} \quad (3.5)$$

$$\psi_{m,k,j}^{(s+1)} = \frac{1 + \alpha_{u,m,k,j}}{2} \psi_{m,k,j}^{u+, (s+1)} + \frac{1 - \alpha_{u,m,k,j}}{2} \psi_{m,k,j}^{u-, (s+1)} \quad (3.6)$$

To implement the *P-PI-SI* iterative method, Eqs. (3.1) - (3.3) are solved for each sub-domain simultaneously by constructing and solving the *ITMM* operators associated with the local system of equations. The process by which these matrices are constructed is discussed later in this chapter. The cell-averaged scalar flux,  $\phi_{k,j}^{(s+\frac{1}{2})}$ , obtained in the *PBJ-ITMM* stage is used to solve Eqs. (3.5) and (3.6) for the cell edge angular fluxes,  $\psi_{m,k,j}^{x+, (s+1)}$  and  $\psi_{m,k,j}^{y+, (s+1)}$ . These equations are solved globally using a single mesh sweep, where the incoming angular fluxes to a spatial cell are obtained either by boundary conditions or from the outgoing angular fluxes of an upstream neighboring cell. Upon conclusion of one *P-PI-SI* iteration, the cell-averaged scalar flux in all cells is tested for relative convergence against a user-set criterion, and the iterative process repeats until a maximum number of iterations is reached, or until convergence is achieved.

Formulating *P-IP-SI*, only one equation in our sequence changes, Eq. (3.1). To change the first step from *PBJ-ITMM* to *IPBJ*, we lag the scalar flux in the scattering source to obtain the following expression from [4], reiterated from the literature review with updated iteration indices to reflect the two-step method.

$$\begin{aligned} v_{x,m,k,j} \left( \psi_{m,k,j}^{x+, (s+\frac{1}{2})} - \psi_{m,k,j}^{x-, (s)} \right) + v_{y,m,k,j} \left( \psi_{m,k,j}^{y+, (s+\frac{1}{2})} - \psi_{m,k,j}^{y-, (s)} \right) + \psi_{m,k,j}^{(s+\frac{1}{2})} \\ = \frac{c_{k,j}}{4\pi} \phi_{k,j}^{(s)} + \frac{1}{4\pi \Sigma_{t,k,j}} q_{k,j} \end{aligned} \quad (3.7)$$

The *IPBJ* step is then the solution of Eqs. (3.2), (3.3), and (3.7) which produce the cell-averaged scalar flux required by the *SI* step. Unlike *PBJ-ITMM*, *IPBJ* does not require a matrix solution. The lagged scalar flux in the scattering source allows a local mesh sweep to be used for the sub-domain solution, analogous to the mesh sweep used for *SI*, but on a local scale.

Following the *IPBJ* step, the *SI* step progresses identically to the *SI* step in *P-PI-SI*, but with one additional equation. The lagging of the cell-averaged scalar flux in the scattering source in *IPBJ* means that this quantity must be calculated in the *SI* step when implementing *P-IP-SI* (this was not required with *P-PI-SI*). This is accomplished with the following equation.

$$\phi_{k,j}^{(s+1)} = \sum_{m=1}^M w_m \psi_{m,k,j}^{(s+1)} \quad (3.8)$$

We also note that while the *SI* step does not have to calculate the cell-averaged scalar flux when using *P-PI-SI*, it is still computed in practice as this is the quantity that is used to test iterative convergence. It is suppressed in the *P-PI-SI* mathematical formulation though, as it is not strictly necessary for the following step in the iterative process.

### 3.1.2 *P-PI-SI* Spectral Analysis

To theoretically analyze the iterative performance of *P-PI-SI*, we employ a Fourier analysis. For this analysis, we consider an infinite homogeneous medium. We then use this configuration to determine the scaling of the iterative error's eigenmodes that results from execution of the combined iterative sequence. An infinite homogeneous (or at least periodic) medium is required in order to perform a Fourier analysis due to the periodicity of the Fourier *ansatz*, on which we project the iterative error. In the theoretical analysis we determine the scaling of the error in the cell-averaged scalar flux due to *SI* stage followed by *PBJ-ITMM* stage, which is the reverse ordering of the implementation of the two methods within a single hybrid iteration as previously described. We reverse the order so that we may solve for the scaling of the error in the cell-averaged scalar flux, which is the quantity that is input into each iteration when *SI* is run first. In the formulation of *P-PI-SI*, we designate *PBJ-ITMM* to be run first so as to emphasize that *PBJ-ITMM* is the primary method, and *SI* is being used in a manner akin to an acceleration method.

To perform this Fourier analysis, we begin by subtracting Eqs. (3.5) and (3.6) from their converged counterparts, decrementing the iteration indices by  $\frac{1}{2}$ , to obtain,

$$\begin{aligned} v_{x,m,k,j} \left( \delta\psi_{m,k,j}^{x+,(s+\frac{1}{2})} - \delta\psi_{m,k,j}^{x-,(s+\frac{1}{2})} \right) + v_{y,m,k,j} \left( \delta\psi_{m,k,j}^{y+,(s+\frac{1}{2})} - \delta\psi_{m,k,j}^{y-,(s+\frac{1}{2})} \right) \\ + \delta\psi_{m,k,j}^{(s+\frac{1}{2})} = \frac{c_{k,j}}{4\pi} \delta\phi_{k,j}^{(s)} \end{aligned} \quad (3.9)$$

and

$$\delta\psi_{m,k,j}^{(s+\frac{1}{2})} = \frac{1 + \alpha_{u,m,k,j}}{2} \delta\psi^{u+, (s+\frac{1}{2})} + \frac{1 - \alpha_{u,m,k,j}}{2} \delta\psi^{u-, (s+\frac{1}{2})} \quad (3.10)$$

where,

$$\delta\psi_{m,k,j}^{(s+\frac{1}{2})} \equiv \psi_{m,k,j}^{(\infty)} - \psi_{m,k,j}^{(s+\frac{1}{2})}, \quad (3.11)$$

with analogous definitions for the other dependent variables in Eqs. (3.9) and (3.10).

Note that all iteration indices in these equations are  $\frac{1}{2}$  smaller than in the *SI* equations of the method formulation to reflect that *SI* is performed first in the Fourier analysis. We now introduce the following Fourier *ansatz*,

$$\delta\phi_{k,j}^{(s)} = \Phi^{(s)}(\lambda_x, \lambda_y) \exp\left(i\Sigma_t \left(\lambda_x \left(k - \frac{1}{2}\right) \Delta x + \lambda_y \left(j - \frac{1}{2}\right) \Delta y\right)\right), \quad (3.12)$$

$$\delta\psi_{m,k,j}^{(s+\frac{1}{2})} = \Psi_m^{(s+\frac{1}{2})}(\lambda_x, \lambda_y) \exp\left(i\Sigma_t \left(\lambda_x \left(k - \frac{1}{2}\right) \Delta x + \lambda_y \left(j - \frac{1}{2}\right) \Delta y\right)\right), \quad (3.13)$$

and

$$\begin{aligned} \delta\psi_{m,k,j}^{u\pm, (s+\frac{1}{2})} &= \Psi_m^{u, (s+\frac{1}{2})}(\lambda_x, \lambda_y) \\ &\times \exp\left(i\Sigma_t \left(\lambda_x \left(k - \frac{1}{2}\right) \Delta x + \lambda_y \left(j - \frac{1}{2}\right) \Delta y \pm \frac{\lambda_u sg(\Omega_{u,m}) \Delta u}{2}\right)\right). \end{aligned} \quad (3.14)$$

In these equations,  $\Phi(\lambda_x, \lambda_y)$ ,  $\Psi_m(\lambda_x, \lambda_y)$ , and  $\Psi_m^u(\lambda_x, \lambda_y)$  represent the magnitude of the iterative error in the cell-averaged scalar flux, cell-averaged angular flux, and cell-edge angular flux on the  $u$  face of the cell, respectively, at the specified frequencies denoted by  $(\lambda_x, \lambda_y)$ . Additionally,  $sg$  in the *signum* function, and  $\Delta x$  and  $\Delta y$  are the  $x$  and  $y$  dimensions of the cell, respectively.

We now substitute these Fourier *ansatz*, Eqs. (3.13) and (3.14), into the residual auxiliary equation, Eq. (3.10), solve for  $\Psi_m^{(s+\frac{1}{2})}(\lambda_x, \lambda_y)$ , employ Euler's formula, and simplify to obtain the following expression.

$$\Psi_m^{(s+\frac{1}{2})}(\lambda_x, \lambda_y) = \Psi_m^{u,(s+\frac{1}{2})}(\lambda_x, \lambda_y) \cos(\rho_{u,m}) + i \alpha_{u,m} \Psi_m^{u,(s+\frac{1}{2})}(\lambda_x, \lambda_y) \sin(\rho_{u,m}) \quad (3.15)$$

where for simplicity, we define,

$$\rho_{u,m} \equiv \frac{\lambda_u s g(\Omega_{u,m}) \Delta u \Sigma_t}{2}. \quad (3.16)$$

We now substitute all three *ansatz*, Eqs. (3.12), (3.13), and (3.14), into the residual balance equation, Eq. (3.9). As we do this, we use the expression for  $\Psi_m^{(s+\frac{1}{2})}(\lambda_x, \lambda_y)$  we just obtained as well as Euler's formula to obtain the following equation after simplifying.

$$\begin{aligned} & \Psi_m^{u,(s+\frac{1}{2})}(\lambda_x, \lambda_y) \cos(\rho_{u,m}) + \quad (3.17) \\ & i \left[ 2 \left( v_{u,m} \Psi_m^{u,(s+\frac{1}{2})}(\lambda_x, \lambda_y) \sin(\rho_{u,m}) + v_{v,m} \Psi_m^{v,(s+\frac{1}{2})}(\lambda_x, \lambda_y) \sin(\rho_{v,m}) \right) \right. \\ & \quad \left. + \Psi_m^{u,(s+\frac{1}{2})}(\lambda_x, \lambda_y) \alpha_{u,m} \sin(\rho_{u,m}) \right] \\ & = \frac{c}{4\pi} \Phi^{(s)}(\lambda_x, \lambda_y), \quad v \neq u = x \text{ or } y \end{aligned}$$

In order to eliminate the  $\Psi_m^{v,(s+\frac{1}{2})}(\lambda_x, \lambda_y)$  term from this equation, we take into account that the left-hand-side of the residual auxiliary equation, Eq. (3.10), does not depend on the choice of  $u$ . Therefore, we equate the right-hand-side of Eq. (3.10) with superscript  $u$  to the right-hand-side with superscript  $v$ . We then substitute the *ansatz*, Eq. (3.14) and solve for  $\Psi_m^{v,(s+\frac{1}{2})}(\lambda_x, \lambda_y)$  to obtain,

$$\Psi_m^{v,(s+\frac{1}{2})}(\lambda_x, \lambda_y) = \beta_m^{u,v} \Psi_m^{u,(s+\frac{1}{2})}(\lambda_x, \lambda_y), \quad (3.18)$$

where we define,

$$\beta_m^{u,v} \equiv \frac{\cos(\rho_{u,m}) + i\alpha_{u,m} \sin(\rho_{u,m})}{\cos(\rho_{v,m}) + i\alpha_{v,m} \sin(\rho_{v,m})}. \quad (3.19)$$

For brevity, we omit the dependence of  $\beta_m^{u,v}$  and  $\rho_{u,m}$  on  $(\lambda_x, \lambda_y)$  in these equations. We now substitute Eq. (3.18) into Eq. (3.17) to eliminate  $\Psi_m^{v,(s+\frac{1}{2})}(\lambda_x, \lambda_y)$ . With this term eliminated, we solve the resulting equation to obtain the following expression that represents the iterative error eigenmodes in the outgoing angular fluxes as a function of the previous iterate's error eigenmodes in the cell-averaged scalar flux,

$$\Psi_m^{u,(s+\frac{1}{2})}(\lambda_x, \lambda_y) = \chi_m^{u,v} \Phi^{(s)}(\lambda_x, \lambda_y) \quad (3.20)$$

where we define,

$$\chi_m^{u,v} \equiv \frac{\frac{c}{4\pi}}{\cos(\rho_{u,m}) + i[2(\nu_{u,m} \sin(\rho_{u,m}) + \nu_{v,m} \beta_m^{u,v} \sin(\rho_{v,m})) + \alpha_{u,m} \sin(\rho_{u,m})]}. \quad (3.21)$$

With an expression for the scaling of the iterative error due to the *SI* step, we now examine the *PBJ-ITMM* step. We conduct the Fourier analysis of the *PBJ-ITMM* using the formalism reported in [4]. Our analysis will differ from [4] when we combine the analysis with the previous *SI* Fourier analysis to obtain the reduction in error from the combined iterative sequence. Also, in [4] the authors obtained the angular dependence of the error, whereas we are only interested in the reduction in error of the scalar flux, as this is the quantity which will be passed to the *SI* step. As with the *SI* step, we begin by subtracting the *PBJ-ITMM* equations, Eqs. (3.1) - (3.3), from their converged counterparts, increasing the iteration indices by  $\frac{1}{2}$  to account for the fact that the *PBJ-ITMM* step is being executed second in the iterative sequence.

$$\begin{aligned}
v_{x,m,k,j} \left( \delta\psi_{m,k,j}^{x+,(s+1)} - \delta\psi_{m,k,j}^{x-,(s+\frac{1}{2})} \right) + v_{y,m,k,j} \left( \delta\psi_{m,k,j}^{y+,(s+1)} - \delta\psi_{m,k,j}^{y-,(s+\frac{1}{2})} \right) + \delta\psi_{m,k,j}^{(s+1)} \quad (3.22) \\
= \frac{c_{k,j}}{4\pi} \delta\phi_{k,j}^{(s+1)}
\end{aligned}$$

$$\delta\psi_{m,k,j}^{(s+1)} = \frac{1 + \alpha_{u,m,k,j}}{2} \delta\psi^{u+,(s+1)} + \frac{1 - \alpha_{u,m,k,j}}{2} \delta\psi^{u-,(s+\frac{1}{2})} \quad (3.23)$$

$$\delta\phi_{k,j}^{(s+1)} = \sum_{m=1}^M w_m \delta\psi_{m,k,j}^{(s+1)} \quad (3.24)$$

The residual auxiliary equation, Eq. (3.23), is solved for the outgoing angular flux residual,  $\delta\psi_{m,k,j}^{u+,(s+1)}$ , and the resulting expression is substituted into the residual balance equation, Eq. (3.22), to obtain,

$$\delta\psi_{m,k,j}^{(s+1)} = \frac{\frac{c_{k,j}}{4\pi} \delta\phi_{k,j}^{(s+1)} + \zeta_{x,m,k,j} \delta\psi_{m,k,j}^{x-,(s+\frac{1}{2})} + \zeta_{y,m,k,j} \delta\psi_{m,k,j}^{y-,(s+\frac{1}{2})}}{1 + \zeta_{x,m,k,j} + \zeta_{y,m,k,j}} \quad (3.25)$$

where,

$$\zeta_{u,m,k,j} \equiv \frac{2v_{u,m,k,j}}{1 + \alpha_{u,m,k,j}}. \quad (3.26)$$

Multiplying by  $w_m$  and summing over all directions, utilizing the quadrature formula in Eq. (3.24), produces the following expression after simplification.

$$\delta\phi_{k,j}^{(s+1)} = \sum_{m=1}^M \left( \gamma_{m,k,j}^x \delta\psi_{m,k,j}^{x-,(s+\frac{1}{2})} + \gamma_{m,k,j}^y \delta\psi_{m,k,j}^{y-,(s+\frac{1}{2})} \right) \quad (3.27)$$

where,

$$\gamma_{m,k,j}^u \equiv w_m \eta_{k,j} \frac{\zeta_{u,m,k,j}}{1 + \zeta_{x,m,k,j} + \zeta_{y,m,k,j}} \quad (3.28)$$

and

$$\eta_{k,j} \equiv \left( 1 - \frac{c_{k,j}}{4\pi} \sum_{m=1}^M w_m \frac{1}{1 + \zeta_{x,m,k,j} + \zeta_{y,m,k,j}} \right)^{-1}. \quad (3.29)$$

We now introduce the following *ansatz* into the *PBJ-ITMM* step.

$$\delta\phi_{k,j}^{(s+1)} = \Phi^{(s+1)}(\lambda_x, \lambda_y) \exp\left(i\Sigma_t \left(\lambda_x \left(k - \frac{1}{2}\right) \Delta x + \lambda_y \left(j - \frac{1}{2}\right) \Delta y\right)\right) \quad (3.30)$$

$$\begin{aligned} \delta\psi_{m,k,j}^{u-, (s+\frac{1}{2})} &= \Psi_m^{u, (s+\frac{1}{2})}(\lambda_x, \lambda_y) \quad (3.31) \\ &\times \exp\left(i\Sigma_t \left(\lambda_x \left(k - \frac{1}{2}\right) \Delta x + \lambda_y \left(j - \frac{1}{2}\right) \Delta y - \frac{\lambda_u s g(\Omega_{u,m}) \Delta u}{2}\right)\right) \end{aligned}$$

We substitute these two expressions into Eq. (3.27) and simplify to obtain,

$$\begin{aligned} &\Phi^{(s+1)}(\lambda_x, \lambda_y) \quad (3.32) \\ &= \sum_{m=1}^M \left[ \gamma_m^x \Psi_m^{x, (s+\frac{1}{2})}(\lambda_x, \lambda_y) \exp(-i\rho_{x,m}) + \gamma_m^y \Psi_m^{y, (s+\frac{1}{2})}(\lambda_x, \lambda_y) \exp(-i\rho_{y,m}) \right]. \end{aligned}$$

Recall that from the *SI* step we obtained an expression for  $\Psi_m^{u, (s+\frac{1}{2})}(\lambda_x, \lambda_y)$ , Eq. (3.20). Therefore, to obtain the scaling of the iterative error in the cell-averaged scalar flux after the combined iterative sequence, we substitute this expression into the previous equation and divide by  $\Phi^{(s)}(\lambda_x, \lambda_y)$  to obtain,

$$\frac{\Phi^{(s+1)}(\lambda_x, \lambda_y)}{\Phi^{(s)}(\lambda_x, \lambda_y)} = \sum_{m=1}^M [\gamma_m^x \chi_m^{x,y} \exp(-i\rho_{x,m}) + \gamma_m^y \chi_m^{y,x} \exp(-i\rho_{y,m})]. \quad (3.33)$$

Equation (3.33) is the final result of the Fourier analysis of *P-PI-SI*, providing the scaling of the iterative error in the cell-averaged scalar flux for the combined iterative sequence as a function of the Fourier variables. We will also refer to this expression as the eigenvalues of the *P-PI-SI* operator, as the scaling of the combined iterative sequence represents the eigenvalues of the hybrid iteration operator. Plots obtained from this equation are presented in Sec. 4.2.

### 3.1.3 *P-IP-SI* Spectral Analysis

We analyze *P-IP-SI* using Fourier analysis analogously to that of *P-PI-SI*. Our motivation for this analysis, however, is different. The underlying mechanism which our hybrid approach was founded on was that *SI* (or other sweep-based method) would model the long-distance streaming and a *PBJ* method would resolve local collisions, the dominant transport phenomena in optically thin and thick cells, respectively. Based on this understanding, it was therefore predicted that *IPBJ* would be ill suited as the primary method in our hybrid approach. Fourier analysis, and later testing, of *P-IP-SI* is therefore conducted to confirm the benefit of employing *PBJ-ITMM* as the primary method in our hybrid approach as well as to verify our understanding of the underlying mechanism of this iterative technique.

As with *P-PI-SI*, the Fourier analysis of *P-IP-SI* will be conducted with *SI* as the first step of the two-step iteration for reasoning discussed in the previous section. *IPBJ* lags the incoming angular fluxes and the cell-averaged scalar flux, indicating that we must derive expressions for the scaling of the error in these two quantities due to the *SI* step. The former of these was already obtained during the *P-PI-SI* Fourier analysis, Eq. (3.20). The latter, however, was not required for the *P-PI-SI* analysis and must now be obtained. To derive this expression, we begin by substituting the *ansatz*, Eqs. (3.12), (3.13), and (3.14) into the *SI* residual balance equation, Eq. (3.9), then simplifying to obtain,

$$\begin{aligned}
& \Psi_m^{(s+\frac{1}{2})}(\lambda_x, \lambda_y) + \\
& 2i \left( v_{x,m} \Psi_m^{x,(s+\frac{1}{2})}(\lambda_x, \lambda_y) \sin(\rho_{x,m}) + v_{y,m} \Psi_m^{y,(s+\frac{1}{2})}(\lambda_x, \lambda_y) \sin(\rho_{y,m}) \right) \\
& = \frac{c}{4\pi} \Phi^{(s)}(\lambda_x, \lambda_y).
\end{aligned} \tag{3.34}$$

We then combine the previous equation with Eq. (3.15) to eliminate  $\Psi_m^{u,(s+\frac{1}{2})}(\lambda_x, \lambda_y)$  and then solve for  $\Psi_m^{(s+\frac{1}{2})}(\lambda_x, \lambda_y)$ ,

$$\begin{aligned}
\Psi_m^{(s+\frac{1}{2})}(\lambda_x, \lambda_y) &= \frac{c}{4\pi} \Phi^{(s)}(\lambda_x, \lambda_y) \times \\
& \left[ 1 + 2i \left( \frac{v_{x,m} \sin(\rho_{x,m})}{\cos(\rho_{x,m}) + i\alpha_{x,m} \sin(\rho_{x,m})} + \frac{v_{y,m} \sin(\rho_{y,m})}{\cos(\rho_{y,m}) + i\alpha_{y,m} \sin(\rho_{y,m})} \right) \right]^{-1}.
\end{aligned} \tag{3.35}$$

This expression calculates the iterative error in the cell-averaged angular flux after the *SI* step in terms of the error in the cell-averaged scalar flux before the *SI* step. To calculate the scaling in the cell-averaged scalar flux, we subtract Eq. (3.8) from its converged counterpart to obtain the following equation, noting that the iteration index has  $\frac{1}{2}$  subtracted from its original value to reflect *SI* being performed first in the Fourier analysis.

$$\delta\phi_{k,j}^{(s+\frac{1}{2})} = \sum_{m=1}^M w_m \delta\psi_{m,k,j}^{(s+\frac{1}{2})} \tag{3.36}$$

where,

$$\delta\phi_{k,j}^{(s+\frac{1}{2})} \equiv \phi_{k,j}^{(\infty)} - \phi_{k,j}^{(s+\frac{1}{2})}. \tag{3.37}$$

We then introduce the following additional *ansatz*.

$$\delta\phi_{k,j}^{(s+\frac{1}{2})} = \Phi^{(s+\frac{1}{2})}(\lambda_x, \lambda_y) \exp\left(i\Sigma_t\left(\lambda_x\left(k - \frac{1}{2}\right)\Delta x + \lambda_y\left(j - \frac{1}{2}\right)\Delta y\right)\right) \quad (3.38)$$

This *ansatz* is substituted into Eq. (3.36) to obtain the relation between  $\Psi_m^{(s+\frac{1}{2})}(\lambda_x, \lambda_y)$  and  $\Phi^{(s+\frac{1}{2})}(\lambda_x, \lambda_y)$ ,

$$\Phi^{(s+\frac{1}{2})}(\lambda_x, \lambda_y) = \sum_{m=1}^M w_m \Psi_m^{(s+\frac{1}{2})}(\lambda_x, \lambda_y). \quad (3.39)$$

We then multiply Eq. (3.35) by  $w_m$  and sum over angle, using the previous relation to obtain,

$$\begin{aligned} \Phi^{(s+\frac{1}{2})}(\lambda_x, \lambda_y) &= \frac{c}{4\pi} \Phi^{(s)}(\lambda_x, \lambda_y) \sum_{m=1}^M w_m \times \\ &\left[ 1 + 2i \left( \frac{v_{x,m} \sin(\rho_{x,m})}{\cos(\rho_{x,m}) + i\alpha_{x,m} \sin(\rho_{x,m})} + \frac{v_{y,m} \sin(\rho_{y,m})}{\cos(\rho_{y,m}) + i\alpha_{y,m} \sin(\rho_{y,m})} \right) \right]^{-1}. \end{aligned} \quad (3.40)$$

This expression calculates the error in the cell-averaged scalar flux after the *SI* step of *P-IP-SI* given the error in that same quantity before the *SI* step. This provides the additional expression that is needed to analyze the *IPBJ* step, thus leaving Eqs. (3.20) and (3.40) as the final results of the analysis of the *SI* step that we use in subsequent analysis.

To continue our Fourier analysis of *P-IP-SI*, we advance to the *IPBJ* step, analysis of which begins by subtracting the *IPBJ* equations, Eqs. (3.2), (3.3), and (3.7) from their converged counterparts to obtain,

$$\begin{aligned} v_{x,m,k,j} \left( \delta\psi_{m,k,j}^{x+, (s+1)} - \delta\psi_{m,k,j}^{x-, (s+\frac{1}{2})} \right) + v_{y,m,k,j} \left( \delta\psi_{m,k,j}^{y+, (s+1)} - \delta\psi_{m,k,j}^{y-, (s+\frac{1}{2})} \right) + \delta\psi_{m,k,j}^{(s+1)} \\ = \frac{c_{k,j}}{4\pi} \delta\phi_{k,j}^{(s+\frac{1}{2})}, \end{aligned} \quad (3.41)$$

$$\delta\psi_{m,k,j}^{(s+1)} = \frac{1 + \alpha_{u,m,k,j}}{2} \delta\psi_{m,k,j}^{u+, (s+1)} + \frac{1 - \alpha_{u,m,k,j}}{2} \delta\psi_{m,k,j}^{u-, (s+\frac{1}{2})}, \quad u = x \text{ or } y, \quad (3.42)$$

and

$$\delta\phi_{k,j}^{(s+1)} = \sum_{m=1}^M w_m \delta\psi_{m,k,j}^{(s+1)}. \quad (3.43)$$

Note that the iteration indices have been increased by  $\frac{1}{2}$  in these equations to denote the fact that *IPBJ* is the second step in our Fourier analysis instead of the first. The residual terms in these equations are defined analogously to Eq. (3.11). We now solve for  $\delta\phi_{k,j}^{(s+1)}$ , the derivation of which directly follows the previously reported analysis of *IPBJ*. [4] As with *P-PI-SI*, our analysis will diverge from that of [4] when we combine the results from both of the steps. To obtain the aforementioned expression, we combine the residual auxiliary equations with the residual balance equation for the *IPBJ* step, Eqs. (3.42) and (3.41) respectively, and solve for the cell-averaged angular flux residual,

$$\delta\psi^{(s+1)} = \frac{\frac{c_{k,j}}{4\pi} \delta\phi_{k,j}^{(s+\frac{1}{2})} + \zeta_{x,m,k,j} \delta\psi_{m,k,j}^{x-, (s+\frac{1}{2})} + \zeta_{y,m,k,j} \delta\psi_{m,k,j}^{y-, (s+\frac{1}{2})}}{1 + \zeta_{x,m,k,j} + \zeta_{y,m,k,j}}. \quad (3.44)$$

Multiplying this equation by  $w_m$ , summing over angle, and applying the quadrature summation for the *IPBJ* step in the Fourier analysis, Eq. (3.43), results in the following expression.

$$\begin{aligned} \delta\phi_{k,j}^{(s+1)} &= \frac{c_{k,j}}{4\pi} \sum_m \frac{w_m}{1 + \zeta_{x,m,j,k} + \zeta_{y,m,j,k}} \delta\phi_{k,j}^{(s+\frac{1}{2})} \\ &+ \sum_{m'} \frac{w'_m \zeta_{x,m',k,j}}{1 + \zeta_{x,m',j,k} + \zeta_{y,m',j,k}} \delta\psi_{m',k,j}^{x-, (s+\frac{1}{2})} \\ &+ \sum_{m''} \frac{w''_m \zeta_{y,m'',k,j}}{1 + \zeta_{x,m'',j,k} + \zeta_{y,m'',j,k}} \delta\psi_{m'',k,j}^{y-, (s+\frac{1}{2})} \end{aligned} \quad (3.45)$$

In this equation,  $m'$  and  $m''$  are simply extra angular indices used to distinguish the independent summations. We then substitute the Fourier *ansatz*, Eqs. (3.30), (3.31), and (3.38) into this expression, transforming it to the frequency domain.

$$\begin{aligned}
\Phi^{(s+1)}(\lambda_x, \lambda_y) & \tag{3.46} \\
&= \frac{c}{4\pi} \left( \sum_m \frac{w_m}{1 + \zeta_{x,m} + \zeta_{y,m}} \right) \Phi^{(s+\frac{1}{2})}(\lambda_x, \lambda_y) \\
&+ \sum_{m'} \frac{w'_m \zeta_{x,m'}}{1 + \zeta_{x,m'} + \zeta_{y,m'}} \Psi_{m'}^{x,(s+\frac{1}{2})}(\lambda_x, \lambda_y) \exp(-i\rho_{x,m'}) \\
&+ \sum_{m''} \frac{w''_m \zeta_{y,m''}}{1 + \zeta_{x,m''} + \zeta_{y,m''}} \Psi_{m''}^{y,(s+\frac{1}{2})}(\lambda_x, \lambda_y) \exp(-i\rho_{y,m''})
\end{aligned}$$

To complete our analysis, recall that we obtained expressions for  $\Phi^{(s+\frac{1}{2})}(\lambda_x, \lambda_y)$  and  $\Psi_m^{u,(s+\frac{1}{2})}(\lambda_x, \lambda_y)$  during analysis of the *SI* step, Eqs. (3.20) and (3.40), respectively. Substituting these two expressions into the previous equation and simplifying yields the final result of our Fourier analysis of *P-IP-SI*,

$$\begin{aligned}
\frac{\Phi^{(s+1)}(\lambda_x, \lambda_y)}{\Phi^{(s)}(\lambda_x, \lambda_y)} &= \left( \frac{c}{4\pi} \right)^2 \sum_m \frac{w_m}{1 + \zeta_{x,m} + \zeta_{y,m}} \times \tag{3.47} \\
&\sum_{m'} \left[ w_{m'} \left( 1 + \frac{2i\nu_{x,m'} \sin(\rho_{x,m'})}{\cos(\rho_{x,m'}) + i\alpha_{x,m'} \sin(\rho_{x,m'})} \right. \right. \\
&\quad \left. \left. + \frac{2i\nu_{y,m'} \sin(\rho_{y,m'})}{\cos(\rho_{y,m'}) + i\alpha_{y,m'} \sin(\rho_{y,m'})} \right)^{-1} \right] \\
&+ \sum_{m''} \frac{w_{m''} \zeta_{x,m''}}{1 + \zeta_{x,m''} + \zeta_{y,m''}} \chi_{m''}^{x,y} \exp(-i\rho_{x,m''}) \\
&+ \sum_{m'''} \frac{w_{m'''} \zeta_{y,m'''}}{1 + \zeta_{x,m'''} + \zeta_{y,m'''}} \chi_{m'''}^{y,x} \exp(-i\rho_{y,m'''}).
\end{aligned}$$

In this equation,  $m'''$  is simply another angular index used to indicate separate summations. The previous equation is the final result of the Fourier analysis of  $P$ - $IP$ - $SI$ , computing the scaling of the iterative error in the cell-averaged scalar flux due to the execution of a single  $P$ - $IP$ - $SI$  iteration. Plots obtained from this equation are presented in Sec. 4.2.

## 3.2 *ITMM* Matrices Construction Algorithms

Performing transport calculations with  $PBJ$ - $ITMM$  requires an algorithm for constructing the response matrices that solve the transport equation locally over a sub-domain to obtain cell-averaged scalar fluxes and the outgoing angular fluxes at sub-domain boundaries. Considering a sub-domain to be an independent transport sub-problem, these matrices must couple all boundary incoming angular and cell-averaged scalar fluxes (assuming isotropic scattering) to all boundary outgoing angular and cell-averaged scalar fluxes in the subject sub-problem. In the construction algorithms we discuss, the objective is to calculate the coupling of the latter quantities to the former under the condition of a lagged scattering source, even though  $ITMM$  does not lag this quantity. By initially lagging it though, the construction algorithms are executed using sweep algorithms akin to those used to execute  $SI$  due to the decoupling of angles. Taking the matrix representation of the transport equation to convergence then allows for the obtained matrices to be used to solve for the converged local transport solution.

In this section, we introduce the matrix equations which  $ITMM$  solves along with two algorithms for the construction of the required matrices. These algorithms are the previously reported differential mesh sweep ( $DMS$ ), developed by *Azmy* [53] and our newly developed Green's Function [1] scheme. The differential mesh sweep is presented by *Zerr* in [52] for 3-D Cartesian geometry. We repeat this derivation for 2-D geometry, as this is the dimensionality of our transport test code, HAT-2C. We then introduce our new approach with the objective of supplying an algorithm with similar construction cost to  $DMS$  that is suited for use on unstructured meshes. This algorithm utilizes the interpretation of  $ITMM$  as a response matrix method, imposing fluxes of unit strength onto the mesh, then executing a single mesh sweep to obtain the response in all other dependent quantities in the sub-problem. The utilization of the mesh sweep algorithm which is traditionally used for solution with  $SI$  facilitates  $ITMM$  construction in unstructured grids, as it uses a kernel calculation and sweep algorithm already present in THOR and other codes.

### 3.2.1 *ITMM* Matrix Equation Formulation

The *ITMM* matrix equations have been previously formulated. [52] We follow the steps in this derivation for a 2-D problem. Additionally, the derivation in [52] was performed for *DD*, while we will implement *AHOT-N0* (or more generally, *WDD*), although this changes very little in the derivation. For this derivation, we treat a sub-domain as a full, independent transport problem. This treatment is valid due to the asynchronicity of *PBJ*, which decouples sub-domains over the course of a single iteration. The construction process described below is performed separately in each sub-domain, hence it readily lends itself to parallel processing. We begin by considering the *SI* transport equation, Eq. (2.73) and the associated auxiliary equations for *AHOT-N0*. These, for a single spatial cell / angle combination can be represented in matrix notation as,

$$\begin{aligned}
 & \begin{bmatrix} 1 & v_{y,m,k,j} & v_{x,m,k,j} \\ 1 & -\frac{1 + \alpha_{y,m,k,j}}{2} & 0 \\ 1 & 0 & -\frac{1 + \alpha_{x,m,k,j}}{2} \end{bmatrix} \begin{bmatrix} \psi_{m,k,j}^{(s+1)} \\ \psi_{m,k,j}^{y+,(s+1)} \\ \psi_{m,k,j}^{x+,(s+1)} \end{bmatrix} \\
 & = \begin{bmatrix} 1 & v_{y,m,k,j} & v_{x,m,k,j} \\ 1 & \frac{1 - \alpha_{y,m,k,j}}{2} & 0 \\ 1 & 0 & \frac{1 - \alpha_{x,m,k,j}}{2} \end{bmatrix} \begin{bmatrix} c_{k,j} \phi_{k,j}^{(s)} + \frac{q_{k,j}}{\Sigma_{t,k,j}} \\ \psi_{m,k,j}^{y-,(s+1)} \\ \psi_{m,k,j}^{x-,(s+1)} \end{bmatrix}.
 \end{aligned} \tag{3.48}$$

This matrix equation is written such that all the inputs to a cell's calculation during a mesh sweep are on the right-hand-side and all quantities that are outputs are on the left-hand-side. Additionally, we see from this matrix equation that the general *WDD* form used is trivial to extend to the more specific cases of *DD* or *AHOT-N0*, with the choice of the weighting factors not impacting the structure of the matrices. Multiplying each side of this equation by the inverse of the coefficient matrix on the left-hand-side yields the following matrix equation which directly maps the inputs for a single cell / angle combination to the outputs,

$$\begin{bmatrix} \psi_{m,k,j}^{(s+1)} \\ \psi_{m,k,j}^{y+, (s+1)} \\ \psi_{m,k,j}^{x+, (s+1)} \end{bmatrix} = \Xi_{m,k,j} \begin{bmatrix} c_{k,j} \phi_{k,j}^{(s)} + \frac{q_{k,j}}{\Sigma_{t,k,j}} \\ \psi_{m,k,j}^{y-, (s+1)} \\ \psi_{m,k,j}^{x-, (s+1)} \end{bmatrix} \quad (3.49)$$

where,

$$\Xi_{m,k,j} \equiv \begin{bmatrix} \xi_{m,k,j}^{aa} & \xi_{m,k,j}^{ay} & \xi_{m,k,j}^{ax} \\ \xi_{m,k,j}^{ya} & \xi_{m,k,j}^{yy} & \xi_{m,k,j}^{yx} \\ \xi_{m,k,j}^{xa} & \xi_{m,k,j}^{xy} & \xi_{m,k,j}^{xx} \end{bmatrix}. \quad (3.50)$$

The  $\xi$  terms in this matrix are coupling factors, which indicate the coupling of an output to a specific input. The first superscript on these terms indicates the output quantity and the second indicates the input quantity ( $a$  is the cell-averaged angular flux if it is the first superscript, the scattering source if it is the second, and  $x/y$  is the angular flux on the  $x/y$  face of the cell, outgoing if it is the first superscript, incoming if it is the second). For example,  $\xi_{m,k,j}^{ay}$  would indicate the contribution to the cell-averaged angular flux from the incoming angular flux on the  $y$  face for angle  $m$ , and cell  $(k, j)$ . These terms are calculated analytically with,

$$\begin{bmatrix} \xi_{m,k,j}^{aa} & \xi_{m,k,j}^{ay} & \xi_{m,k,j}^{ax} \\ \xi_{m,k,j}^{ya} & \xi_{m,k,j}^{yy} & \xi_{m,k,j}^{yx} \\ \xi_{m,k,j}^{xa} & \xi_{m,k,j}^{xy} & \xi_{m,k,j}^{xx} \end{bmatrix} \quad (3.51)$$

$$= \begin{bmatrix} 1 & v_{y,m,k,j} & v_{x,m,k,j} \\ 1 & -\frac{1 + \alpha_{y,m,k,j}}{2} & 0 \\ 1 & 0 & -\frac{1 + \alpha_{x,m,k,j}}{2} \end{bmatrix}^{-1} \begin{bmatrix} 1 & v_{y,m,k,j} & v_{x,m,k,j} \\ 1 & \frac{1 - \alpha_{y,m,k,j}}{2} & 0 \\ 1 & 0 & \frac{1 - \alpha_{x,m,k,j}}{2} \end{bmatrix}.$$

An analytical expression for this matrix was obtained for our implementation using Mathematica. [62] Equations (3.48) - (3.51) are the only matrix-vector equations that change when the derivation is performed in 2-D for *AHOT-N0* rather than in 3-D for *DD*. The remaining equations in this subsection are therefore equivalent to those previously reported in [52]. Considering the full sub-

domain with nonzero boundary conditions, we can represent the cell-averaged scalar fluxes of all cells as a vector calculated with,

$$\boldsymbol{\phi}^{(s+1)} = \mathbf{A}(\mathbf{C}\boldsymbol{\phi}^{(s)} + \boldsymbol{\Sigma}_t^{-1}\mathbf{q}) + \mathbf{K}_\phi\boldsymbol{\psi}_{in}. \quad (3.52)$$

$\mathbf{A}$  and  $\mathbf{K}_\phi$  are matrices which consist of  $\xi$  terms,  $\boldsymbol{\Sigma}_t^{-1}$  is a diagonal matrix of the reciprocals of the macroscopic total cross sections,  $\mathbf{C}$  is a diagonal matrix of the scattering ratio,  $\mathbf{q}$  is a vector of the external source,  $\boldsymbol{\psi}_{in}$  is a vector of the incoming angular fluxes on the sub-domain boundary, and  $\boldsymbol{\phi}$  is a vector of the cell-averaged scalar fluxes, either lagged or the current iterate as indicated by the superscript. The matrices  $\mathbf{A}$  and  $\mathbf{K}_\phi$  map the inputs on the right-hand-side of Eq. (3.49) to the outputs on the left-hand-side. Next, we modify this equation to,

$$\boldsymbol{\phi}^{(s+1)} = \mathbf{A}\mathbf{C}\boldsymbol{\phi}^{(s)} + \mathbf{A}\mathbf{C}\boldsymbol{\Sigma}_s^{-1}\mathbf{q} + \mathbf{K}_\phi\boldsymbol{\psi}_{in}. \quad (3.53)$$

To simplify, we define,

$$\mathbf{J}_\phi \equiv \mathbf{A}\mathbf{C}, \quad (3.54)$$

thus resulting in,

$$\boldsymbol{\phi}^{(s+1)} = \mathbf{J}_\phi(\boldsymbol{\phi}^{(s)} + \boldsymbol{\Sigma}_s^{-1}\mathbf{q}) + \mathbf{K}_\phi\boldsymbol{\psi}_{in}. \quad (3.55)$$

$\boldsymbol{\Sigma}_s^{-1}$  is a diagonal matrix of the reciprocals of the macroscopic scattering cross sections. By taking the iterative limit of this equation, we then solve for the vector of fully converged scalar flux over the sub-domain,

$$\boldsymbol{\phi} = (\mathbf{I} - \mathbf{J}_\phi)^{-1}(\mathbf{J}_\phi\boldsymbol{\Sigma}_s^{-1}\mathbf{q} + \mathbf{K}_\phi\boldsymbol{\psi}_{in}). \quad (3.56)$$

This equation produces the cell-averaged scalar flux for each cell in a sub-domain given prescribed incoming angular fluxes on the sub-domain boundary. To implement *PBJ-ITMM*

though, we must also calculate the outgoing angular fluxes at the sub-domain boundary, as these will be exchanged between adjacent sub-domains after an iteration completes. The equation to calculate these values is written in matrix-vector notation as,

$$\boldsymbol{\psi}_{out} = \mathbf{J}_{\psi}(\boldsymbol{\phi} + \boldsymbol{\Sigma}_s^{-1}\mathbf{q}) + \mathbf{K}_{\psi}\boldsymbol{\psi}_{in}. \quad (3.57)$$

$\boldsymbol{\psi}_{out}$  is a vector of the outgoing angular fluxes at the sub-domain boundary and  $\mathbf{J}_{\psi}$  and  $\mathbf{K}_{\psi}$  are response matrices which determine these outgoing angular fluxes based on the  $\xi$  terms. Equations (3.56) and (3.57) are the two equations which are solved in every sub-domain, each iteration when executing *PBJ-ITMM*. Equation (3.56) is solved first, obtaining the cell-averaged scalar fluxes given the incoming angular fluxes at each sub-domain's boundaries which are determined from the previous iterate's solution in adjacent sub-domains, the boundary condition, or the initial guess. Then, these sub-domain boundary incoming angular fluxes are used in conjunction with the obtained cell-averaged scalar fluxes to obtain the sub-domain boundary outgoing angular fluxes using Eq. (3.57), which are then used by adjacent sub-domains in the next iteration.

Solving these equations is dependent upon deriving expressions for the elements of the four response matrices,  $\mathbf{J}_{\phi}$ ,  $\mathbf{K}_{\phi}$ ,  $\mathbf{J}_{\psi}$ , and  $\mathbf{K}_{\psi}$ . The following sections present the derivation of these elements followed by a description of two algorithms for their construction. We note here the dimensions of the matrices for a sub-domain of size  $N_x \times N_y$  with  $M$  angles per angular quadrant in 2-D:  $\mathbf{J}_{\phi}$ :  $N_x N_y \times N_x N_y$ ,  $\mathbf{K}_{\phi}$ :  $N_x N_y \times 4M(N_x + N_y)$ ,  $\mathbf{J}_{\psi}$ :  $4M(N_x + N_y) \times N_x N_y$ , and  $\mathbf{K}_{\psi}$ :  $4M(N_x + N_y) \times 4M(N_x + N_y)$ .

### 3.2.2 *ITMM* Matrix Elements

Implementation of *PBJ-ITMM* requires prescriptions for the elements of the four associated matrices,  $\mathbf{J}_{\phi}$ ,  $\mathbf{K}_{\phi}$ ,  $\mathbf{J}_{\psi}$ , and  $\mathbf{K}_{\psi}$ . Construction of these matrices and factorization of  $(\mathbf{I} - \mathbf{J}_{\phi})$  is implemented as a pre-process, performed only once for each energy group, as the matrix elements do not change during the iterative solution. To obtain expressions for these matrix elements, Eqs. (3.55) and (3.57) are differentiated, providing the four following relations. [52]

$$\frac{\partial \phi^{(s+1)}}{\partial \phi^{(s)}} = J_\phi \quad (3.58)$$

$$\frac{\partial \psi_{out}^{(s+1)}}{\partial \phi^{(s)}} = J_\psi \quad (3.59)$$

$$\frac{\partial \phi^{(s+1)}}{\partial \psi_{in}} = K_\phi \quad (3.60)$$

$$\frac{\partial \psi_{out}^{(s+1)}}{\partial \psi_{in}} = K_\psi \quad (3.61)$$

Note that before differentiating Eq. (3.57), we lagged the scalar flux, thus rendering the outgoing angular flux to be the  $(s+1)$  iterate's solution, rather than the iterative limit. By doing so, prescriptions for the *ITMM* matrices are dependent on the lagged scalar flux, a necessary feature of the construction algorithms. Taking the iterative limit does not change the form of Eq. (3.57), rendering this matrix prescription valid. Also, the incoming boundary angular flux vector,  $\boldsymbol{\psi}_{in}$ , does not have an iteration index, as the boundary conditions are treated as fixed for the *ITMM* construction. Subsequently, cell-incoming angular flux terms will contain an  $(s+1)$  iteration index, which are inferred from the boundary condition for sub-domain boundary cells. Evaluating each of these derivatives at individual locations within the matrices yields the following prescriptions for the elements of the four matrices. [52]

$$j_{\phi,(k,j),(k',j')} = \frac{\partial \phi_{k,j}^{(s+1)}}{\partial \phi_{k',j'}^{(s)}} \quad (3.62)$$

$$j_{\psi,(m,N_x,j),(k',j')} = \frac{\partial \psi_{m,N_x,j}^{x+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} \quad \text{and} \quad j_{\psi,(m,k,N_y),(k',j')} = \frac{\partial \psi_{m,k,N_y}^{y+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} \quad (3.63)$$

$$k_{\phi,(k,j),(m,1,j')} = \frac{\partial \phi_{k,j}^{(s+1)}}{\partial \psi_{m,1,j'}^{x-, (s+1)}} \quad \text{and} \quad k_{\phi,(k,j),(m,k',1)} = \frac{\partial \phi_{k,j}^{(s+1)}}{\partial \psi_{m,k',1}^{y-, (s+1)}} \quad (3.64)$$

$$k_{\psi,(m,N_x,j),(m,1,j')} = \frac{\partial \psi_{m,N_x,j}^{x+, (s+1)}}{\partial \psi_{m,1,j'}^{x-, (s+1)}}, \quad (3.65)$$

$$k_{\psi,(m,k,N_y),(m,1,j')} = \frac{\partial \psi_{m,k,N_y}^{y+, (s+1)}}{\partial \psi_{m,1,j'}^{x-, (s+1)}},$$

$$k_{\psi,(m,N_x,j),(m,k',1)} = \frac{\partial \psi_{m,N_x,j}^{x+, (s+1)}}{\partial \psi_{m,k',1}^{y-, (s+1)}}, \quad \text{and}$$

$$k_{\psi,(m,k,N_y),(m,k',1)} = \frac{\partial \psi_{m,k,N_y}^{y+, (s+1)}}{\partial \psi_{m,k',1}^{y-, (s+1)}}$$

The previous four equations prescribe the elements of the four matrices required for *PBJ-ITMM*. Note that in the last three equations (Eqs. (3.63), (3.64), and (3.65)) the expressions involve angular fluxes on the sub-domain boundary, implying that the indices will change depending on the discrete ordinate octant. For this reason, these equations have multiple manifestations, which change between the  $x$  and  $y$  indices that are fixed to the boundary of the sub-domain. The equations provided are for the case of a direction in the first quadrant. For all other quadrants, the boundary to which an index is assigned is changed to indicate the boundary corresponding to incoming or outgoing directions in the subject quadrant for angular fluxes with superscript  $u-$  or  $u+$ , respectively. For example, in the third quadrant, all  $x$  indices of 1 will change to  $N_x$ , all  $x$  indices of  $N_x$  will change to 1, all  $y$  indices of 1 will change to  $N_y$ , and all  $y$  indices of  $N_y$  will change to 1.

Computing the derivatives on the right-hand-side of Eqs. (3.62) - (3.65), the *ITMM* matrix elements, is the purpose of the algorithms which are discussed in the following two sections.

### 3.2.3 *Differential Mesh Sweep (DMS)*

The differential mesh sweep (*DMS*) is a previously developed algorithm [52, 53] for constructing the four matrices required for obtaining a sub-domain's iterative solution using *PBJ-*

*ITMM*,  $\mathbf{J}_\phi$ ,  $\mathbf{K}_\phi$ ,  $\mathbf{J}_\psi$ , and  $\mathbf{K}_\psi$ , who's formulation this section follows. To begin the *DMS* formulation, following from [52, 53] for our 2-D configuration, we differentiate Eq. (3.49) to obtain the nine derivatives for a single cell / direction combination, as follows.

$$\begin{bmatrix} \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \phi_{k,j}^{(s)}} & \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,k,j}^{y-, (s+1)}} & \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,k,j}^{x-, (s+1)}} \\ \frac{\partial \psi_{m,k,j}^{y+, (s+1)}}{\partial \phi_{k,j}^{(s)}} & \frac{\partial \psi_{m,k,j}^{y+, (s+1)}}{\partial \psi_{m,k,j}^{y-, (s+1)}} & \frac{\partial \psi_{m,k,j}^{y+, (s+1)}}{\partial \psi_{m,k,j}^{x-, (s+1)}} \\ \frac{\partial \psi_{m,k,j}^{x+, (s+1)}}{\partial \phi_{k,j}^{(s)}} & \frac{\partial \psi_{m,k,j}^{x+, (s+1)}}{\partial \psi_{m,k,j}^{y-, (s+1)}} & \frac{\partial \psi_{m,k,j}^{x+, (s+1)}}{\partial \psi_{m,k,j}^{x-, (s+1)}} \end{bmatrix} = \begin{bmatrix} c_{k,j} \xi_{m,k,j}^{aa} & \xi_{m,k,j}^{ay} & \xi_{m,k,j}^{ax} \\ c_{k,j} \xi_{m,k,j}^{ya} & \xi_{m,k,j}^{yy} & \xi_{m,k,j}^{yx} \\ c_{k,j} \xi_{m,k,j}^{xa} & \xi_{m,k,j}^{xy} & \xi_{m,k,j}^{xx} \end{bmatrix} \quad (3.66)$$

This set of derivatives that is computable from Eq. (3.50) provides the building blocks which will be used by the *DMS* algorithm to construct the elements of the matrices  $\mathbf{J}_\phi$ ,  $\mathbf{K}_\phi$ ,  $\mathbf{J}_\psi$ , and  $\mathbf{K}_\psi$ . Henceforth, the elements of Eq. (3.66) are referred to as “core derivatives.”

These matrices are constructed via *DMS* through a nested mesh sweep. For a given direction  $m$ , the outer sweep loops over  $k'$  and  $j'$ , where these variables span the spatial cells in the sub-domain in the  $x$  and  $y$  dimensions, respectively. The inner sweep then starts from the cell  $(k', j')$  and sweeps using the indices  $k$  and  $j$  over all cells in the sub-domain which are downstream from cell  $(k', j')$  along the given direction  $m$ . In each step of the inner sweep, the response of a quantity in cell  $(k, j)$  to a change in a quantity in cell  $(k', j')$  is calculated, providing the corresponding matrix element. A nested sweep is conducted twice for each angle, once to calculate the response due to the lagged cell-averaged scalar flux in each cell, then a second time to calculate the response due to the angular flux incoming to each sub-domain boundary cell. The former of these sweeps constructs the elements for  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ , and the latter constructs the elements for  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$ . These sweeps construct the matrix elements as expressed in Eqs. (3.62) – (3.65) using the core derivatives of Eq. (3.66). The derivatives in Eqs. (3.62) – (3.65) relate the changes in flux quantities within a given cell to those of another cell within the sub-domain. To calculate these using the core derivatives which are differentiated within a common cell, we differentiate recursively as described below.

We begin with the construction of the elements for  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ . To construct these elements, consider an arbitrary cell in the sub-domain which we index  $(k', j')$  and an arbitrary direction  $m$  in the first quadrant. The inner sweep begins at this cell index, indicating that  $(k, j) = (k', j')$ . For this starting cell, the derivative of the cell-averaged angular flux with respect to the lagged cell-averaged scalar flux,  $\frac{\partial \psi_{m,k',j'}^{(s+1)}}{\partial \phi_{k',j'}^{(s)}}$ , is calculated directly using the (1,1) entry of Eq. (3.66). The derivatives of the outgoing angular fluxes with respect to the lagged cell-averaged scalar flux,  $\frac{\partial \psi_{m,k',j'}^{x+,(s+1)}}{\partial \phi_{k',j'}^{(s)}}$  and  $\frac{\partial \psi_{m,k',j'}^{y+,(s+1)}}{\partial \phi_{k',j'}^{(s)}}$ , are computed using the (3,1) and (2,1) entries of Eq. (3.66). Then, the sweep progresses to adjacent downstream cells, where  $(k, j) \neq (k', j')$ . To continue the inner sweep, the derivative of the outgoing angular fluxes in cell  $(k, j)$  with respect to the lagged cell-averaged scalar flux in cell  $(k', j')$  is determined via recursive differentiation using derivatives calculated in adjacent upstream cells to  $(k, j)$ ,

$$\frac{\partial \psi_{m,k,j}^{x+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} = \frac{\partial \psi_{m,k-1,j}^{x+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} \frac{\partial \psi_{m,k,j}^{x+,(s+1)}}{\partial \psi_{m,k,j}^{x-,(s+1)}} + \frac{\partial \psi_{m,k,j-1}^{y+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} \frac{\partial \psi_{m,k,j}^{x+,(s+1)}}{\partial \psi_{m,k,j}^{y-,(s+1)}} \quad (3.67)$$

and

$$\frac{\partial \psi_{m,k,j}^{y+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} = \frac{\partial \psi_{m,k-1,j}^{x+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} \frac{\partial \psi_{m,k,j}^{y+,(s+1)}}{\partial \psi_{m,k,j}^{x-,(s+1)}} + \frac{\partial \psi_{m,k,j-1}^{y+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} \frac{\partial \psi_{m,k,j}^{y+,(s+1)}}{\partial \psi_{m,k,j}^{y-,(s+1)}}. \quad (3.68)$$

In each of these equations, two of the derivatives on the right-hand-side are inferred from adjacent upstream neighbors while the other two are core derivatives from Eq. (3.66). Note that these are for the example of a direction in the first quadrant. For quadrants where the sign of the directional cosine for a dimension is negative, the index corresponding to the adjacent upstream neighbor in that dimension will have a +1 rather than a -1. Also note that if  $k = k'$  or  $j = j'$ ,  $\frac{\partial \psi_{m,k-1,j}^{x+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} = 0$  or

$\frac{\partial \psi_{m,k,j-1}^{y+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} = 0$ , respectively. This recursive differentiation relies on the synchronicity between cells

in a common sub-domain, thus implying  $\psi_{m,k-1,j}^{x+,(s+1)} = \psi_{m,k,j}^{x-,(s+1)}$  and  $\psi_{m,k,j-1}^{y+,(s+1)} = \psi_{m,k,j}^{y-,(s+1)}$ . Equations (3.67) and (3.68) advance the inner sweep of the differential mesh sweep algorithm, with the calculated derivatives allowing these equations to be subsequently evaluated in downstream adjacent neighbors.

At each cell during the sweep however, there is another calculation required for constructing  $\mathbf{J}_\phi$ . The necessary derivative,  $\frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \phi_{k',j'}^{(s)}}$  has thus far only been calculated for the case of  $(k, j) = (k', j')$ , as it is simply a core derivative for this case. When  $(k, j) \neq (k', j')$ , this term must be calculated using,

$$\frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \phi_{k',j'}^{(s)}} = \frac{\partial \psi_{m,k-1,j}^{x+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,k,j}^{x-,(s+1)}} + \frac{\partial \psi_{m,k,j-1}^{y+,(s+1)}}{\partial \phi_{k',j'}^{(s)}} \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,k,j}^{y-,(s+1)}}. \quad (3.69)$$

As with Eqs. (3.67) and (3.68), the right-hand-side of Eq. (3.69) contains two derivatives calculated in the adjacent upstream neighbors and two core derivatives. As the sweep progresses, these  $\frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \phi_{k',j'}^{(s)}}$  values are accumulated. Numerical integration over angle then yields the derivative of the updated cell-averaged scalar flux in cell  $(k, j)$  with respect to the lagged cell-averaged scalar flux in cell  $(k', j')$ , [52, 53]

$$\frac{\partial \phi_{k,j}^{(s+1)}}{\partial \phi_{k',j'}^{(s)}} = \sum_{m=1}^M w_m \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \phi_{k',j'}^{(s)}}. \quad (3.70)$$

Referring to Eq. (3.62), we see that these are the elements of the  $\mathbf{J}_\phi$  matrix. To minimize memory usage during this construction,  $\frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \phi_{k',j'}^{(s)}}$  values need not be stored. Instead,  $\frac{\partial \phi_{k,j}^{(s+1)}}{\partial \phi_{k',j'}^{(s)}}$  may be calculated as a running summation, as the  $\frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \phi_{k',j'}^{(s)}}$  values are not subsequently needed.

The elements of  $\mathbf{J}_\psi$ , Eq. (3.63), are obtained with Eq. (3.67) or (3.68) when cell  $(k, j)$  has a face on the outgoing sub-domain  $x$  or  $y$  boundary respectively (two calculations for a corner cell

with two outgoing faces). The inner sweep, when conducted over all directions for a given cell  $(k',j')$ , constructs one column of  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$  after all cells downstream of  $(k',j')$  have been swept. The outer sweep then increments cell index  $(k',j')$  and this process is repeated until  $k'$  and  $j'$  have swept over all cells in the sub-domain, thus constructing all columns of the matrices  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ .

The matrices  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$  are then constructed in a manner very similar to  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ , but obtaining derivatives with respect to the incoming angular fluxes on sub-domain boundaries rather than lagged cell-averaged scalar fluxes. There are two main consequences of this difference: (1) the outer sweep is over incoming sub-domain boundary faces rather than cells within the sub-domain; and (2) a matrix column is produced for each individual direction for a given incoming sub-domain boundary face as the derivatives are with respect to angular flux, rather than scalar flux. With these distinctions,  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$  are constructed via the differential mesh sweep as detailed below, noting that this construction algorithm is for a direction in the first quadrant. The algorithms for other quadrants are analogous. Lastly, the equations for this algorithm will have two different forms depending on whether the incoming sub-domain boundary face is on the  $x$  or  $y$  boundary, both of which are provided. For the corner cell with two incoming faces, each of these faces initiates its own inner sweep.

We begin with a cell,  $(k',j')$ , that has a face on an incoming sub-domain boundary for direction  $m$ . The inner sweep initiates from cell  $(k,j)$  such that  $(k,j) = (k',j')$ . For this beginning cell, the derivative of the cell-averaged angular flux with respect to its incoming angular flux,

$\frac{\partial \psi_{m,1,j'}^{(s+1)}}{\partial \psi_{m,1,j'}^{x-, (s+1)}}$  or  $\frac{\partial \psi_{m,k',1}^{(s+1)}}{\partial \psi_{m,k',1}^{y-, (s+1)}}$ , is a core derivative, entry (1,3) or (1,2) of Eq. (3.66) for a face on an

incoming sub-domain  $x$  or  $y$  boundary respectively. Then, the derivatives of the cell's outgoing

angular fluxes,  $\frac{\partial \psi_{m,1,j'}^{x+, (s+1)}}{\partial \psi_{m,1,j'}^{x-, (s+1)}}$  and  $\frac{\partial \psi_{m,1,j'}^{y+, (s+1)}}{\partial \psi_{m,1,j'}^{y-, (s+1)}}$  or  $\frac{\partial \psi_{m,k',1}^{x+, (s+1)}}{\partial \psi_{m,k',1}^{y-, (s+1)}}$  and  $\frac{\partial \psi_{m,k',1}^{y+, (s+1)}}{\partial \psi_{m,k',1}^{x-, (s+1)}}$ , with respect to its incoming

angular flux are also core derivatives, entries (3,3) and (2,3) or (3,2) and (2,2) from Eq. (3.66) for a face on the incoming sub-domain  $x$  or  $y$  boundary respectively. Analogously to the construction of  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ , these derivatives provide the starting point for the inner sweep to progress, recursively differentiating. The inner sweep progresses to adjacent downstream cells using,

$$\frac{\partial \psi_{m,k,j}^{x+,(s+1)}}{\partial \psi_{m,1,j'}^{x-,(s+1)}} = \frac{\partial \psi_{m,k-1,j}^{x+,(s+1)}}{\partial \psi_{m,1,j'}^{x-,(s+1)}} \frac{\partial \psi_{m,k,j}^{x+,(s+1)}}{\partial \psi_{m,k,j}^{x-,(s+1)}} + \frac{\partial \psi_{m,k,j-1}^{y+,(s+1)}}{\partial \psi_{m,1,j'}^{x-,(s+1)}} \frac{\partial \psi_{m,k,j}^{x+,(s+1)}}{\partial \psi_{m,k,j}^{y-,(s+1)}} \quad (3.71)$$

and

$$\frac{\partial \psi_{m,k,j}^{y+,(s+1)}}{\partial \psi_{m,1,j'}^{x-,(s+1)}} = \frac{\partial \psi_{m,k-1,j}^{x+,(s+1)}}{\partial \psi_{m,1,j'}^{x-,(s+1)}} \frac{\partial \psi_{m,k,j}^{y+,(s+1)}}{\partial \psi_{m,k,j}^{x-,(s+1)}} + \frac{\partial \psi_{m,k,j-1}^{y+,(s+1)}}{\partial \psi_{m,1,j'}^{x-,(s+1)}} \frac{\partial \psi_{m,k,j}^{y+,(s+1)}}{\partial \psi_{m,k,j}^{y-,(s+1)}} \quad (3.72)$$

or

$$\frac{\partial \psi_{m,k,j}^{x+,(s+1)}}{\partial \psi_{m,k',1}^{y-,(s+1)}} = \frac{\partial \psi_{m,k-1,j}^{x+,(s+1)}}{\partial \psi_{m,k',1}^{y-,(s+1)}} \frac{\partial \psi_{m,k,j}^{x+,(s+1)}}{\partial \psi_{m,k,j}^{x-,(s+1)}} + \frac{\partial \psi_{m,k,j-1}^{y+,(s+1)}}{\partial \psi_{m,k',1}^{y-,(s+1)}} \frac{\partial \psi_{m,k,j}^{x+,(s+1)}}{\partial \psi_{m,k,j}^{y-,(s+1)}} \quad (3.73)$$

and

$$\frac{\partial \psi_{m,k,j}^{y+,(s+1)}}{\partial \psi_{m,k',1}^{y-,(s+1)}} = \frac{\partial \psi_{m,k-1,j}^{x+,(s+1)}}{\partial \psi_{m,k',1}^{y-,(s+1)}} \frac{\partial \psi_{m,k,j}^{y+,(s+1)}}{\partial \psi_{m,k,j}^{x-,(s+1)}} + \frac{\partial \psi_{m,k,j-1}^{y+,(s+1)}}{\partial \psi_{m,k',1}^{y-,(s+1)}} \frac{\partial \psi_{m,k,j}^{y+,(s+1)}}{\partial \psi_{m,k,j}^{y-,(s+1)}} \quad (3.74)$$

Equations (3.71) and (3.72) pertain to a face on an incoming sub-domain  $x$  boundary and Eqs. (3.73) and (3.74) pertain to a face on an incoming sub-domain  $y$  boundary. For any cell,  $(k, j)$ , if  $k = k'$  or  $j = j'$ ,  $\partial \psi_{m,k-1,j}^{x+,(s+1)} = 0$  or  $\partial \psi_{m,k,j-1}^{y+,(s+1)} = 0$ , respectively, for either of the possible denominators to each derivative. Note that the index of 1 which fixes the cell  $(k', j')$  to the sub-domain boundary is still considered a  $k'$  or  $j'$  index. Equations (3.71) - (3.74) each contain two derivatives on the right-hand-side that are known from upstream neighbors and two that are core derivatives, Eq. (3.66).

Analogously to the construction of  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ , this allows the inner sweep to progress sequentially along downstream cells. In each cell, the derivative  $\frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,1,j'}^{x-,(s+1)}}$  or  $\frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,1,j'}^{y-,(s+1)}}$  is computed

using

$$\frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,1,j'}^{x-, (s+1)}} = \frac{\partial \psi_{m,k-1,j}^{x+, (s+1)}}{\partial \psi_{m,1,j'}^{x-, (s+1)}} \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,k,j}^{x-, (s+1)}} + \frac{\partial \psi_{m,k,j-1}^{y+, (s+1)}}{\partial \psi_{m,1,j'}^{x-, (s+1)}} \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,k,j}^{y-, (s+1)}} \quad (3.75)$$

or

$$\frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,k',1}^{y-, (s+1)}} = \frac{\partial \psi_{m,k-1,j}^{x+, (s+1)}}{\partial \psi_{m,k',1}^{y-, (s+1)}} \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,k,j}^{x-, (s+1)}} + \frac{\partial \psi_{m,k,j-1}^{y+, (s+1)}}{\partial \psi_{m,k',1}^{y-, (s+1)}} \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,k,j}^{y-, (s+1)}} \quad (3.76)$$

for a face on an incoming sub-domain  $x$  or  $y$  boundary respectively. Again, the right-hand-sides of each of these equations consist of two derivatives previously calculated in upstream neighbors and two core derivatives.

Referring to Eq. (3.64), due to the lagging of the scattering source, all angles are decoupled, implying that an impingent boundary flux along the  $m^{\text{th}}$  direction will contribute only to angular flux quantities in this same direction. We are therefore able to numerically integrate Eq. (3.64), accounting for all other directions' angular fluxes to be zero, yielding, [52, 53]

$$k_{\phi, (k,j), (m,1,j')} = w_m \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,1,j'}^{x-, (s+1)}} \quad \text{or} \quad k_{\phi, (k,j), (m,k',1)} = w_m \frac{\partial \psi_{m,k,j}^{(s+1)}}{\partial \psi_{m,k',1}^{y-, (s+1)}}. \quad (3.77)$$

Consequently, Eq. (3.75) or (3.76) (depending on  $x$  or  $y$  incoming sub-domain boundary) supplies an element of  $\mathbf{K}_\phi$  for each cell  $(k,j)$  of the inner sweep, calculated using the previous equation.

When the inner sweep reaches a cell which has a face on the outgoing  $x$  or  $y$  boundary, Eq. (3.71), (3.72), (3.73), or (3.74) then provides an element of  $\mathbf{K}_\psi$  as prescribed by Eq. (3.65). After  $(k,j)$  sweeps over all cells downstream of cell  $(k',j')$ , the inner sweep is complete. The outer sweep progresses by simply repeating this same process for all  $(k',j')$  and  $m$  specifications of incoming sub-domain boundary face fluxes.

This concludes the description of the differential mesh sweep algorithm. From this construction algorithm, there are a few properties that we note. All the inner sweeps for the differential mesh sweep only span cells that are downstream from  $(k',j')$ . Any values pertaining to cells upstream of  $(k',j')$  are zero, as the contribution to a quantity in an upstream cell is clearly

zero. Also,  $\mathbf{K}_\psi$  couples sub-domain boundary outgoing angular fluxes in all directions to sub-domain boundary incoming angular fluxes from all directions. However, due to the lagged scattering source there is only non-zero coupling between quantities pertaining to the same angle, and the majority of  $\mathbf{K}_\psi$ 's elements are zero for most cases. Thus, sparse matrix construction of this matrix is advised to reduce its memory footprint.

### 3.2.4 *Green's Function ITMM Construction (GFIC) Algorithm*

Implementation of *DMS* for *PBJ-ITMM* solution on unstructured grids, that is the ultimate goal of this work, raises complications. The construction of  $\mathbf{J}_\phi$ ,  $\mathbf{K}_\phi$ ,  $\mathbf{J}_\psi$ , and  $\mathbf{K}_\psi$  via *DMS* requires the calculation of the core derivatives, Eq. (3.66), which is contingent upon the ability to calculate the  $\xi$  terms using Eq. (3.51). On structured grids, analytical inversion of the matrix on the right-hand-side of this equation is easy, as the cells of the grid have a simple shape even if their dimensions vary. On unstructured grids, however, nonuniformity of the cells' shapes make the counterpart to Eq. (3.51) vary in form between cells, thus complicating analytical inversion.

To circumvent this complication, we develop a novel response matrices construction algorithm, a Green's Function approach. The underlying concept of this approach is to, utilizing the interpretation of the *ITMM* matrices as response matrices, create an artificial set of problem parameters that allow traditional *SI* mesh sweeps to yield the matrix elements. Generally, this is accomplished by selectively imposing unit values to flux quantities within the sub-domain which render the solution obtained from a generic *SI* mesh sweep equivalent to the matrix elements obtained with *DMS*. The use of previously existing code capabilities, namely the mesh sweep algorithm, renders this approach largely agnostic to the mesh type and spatial discretization method, which additionally makes the implementation in production codes minimally intrusive.

The Green's Function *ITMM* Construction (*GFIC*) algorithm capitalizes on the interpretation of *ITMM* as a response matrix method. Considering the *ITMM* matrices to contain elements that represent the change is one quantity given a change in another, we propose an algorithm where these terms are computationally generated by imposing a change to a quantity in a manner such that the resulting change in other quantities is computable. To begin, consider the prescriptions for the elements of the *ITMM* matrices, Eqs. (3.62) - (3.65). Given the linearity of the neutron transport equation, these derivatives may be expressed as quotients of the discrete

changes in the corresponding quantities, producing the following modified prescriptions for the elements of the *ITMM* matrices.

$$j_{\phi,(k,j),(k',j')} = \frac{\Delta\phi_{k,j}^{(s+1)}}{\Delta\phi_{k',j'}^{(s)}} = \frac{\left(\phi_{k,j}^{(s+1)}\right)_{GFIC} - \left(\phi_{k,j}^{(s+1)}\right)_0}{\left(\phi_{k',j'}^{(s)}\right)_{GFIC} - \left(\phi_{k',j'}^{(s)}\right)_0} \quad (3.78)$$

$$j_{\psi,(m,N_x,j),(k',j')} = \frac{\Delta\psi_{m,N_x,j}^{x+,(s+1)}}{\Delta\phi_{k',j'}^{(s)}} = \frac{\left(\psi_{m,N_x,j}^{x+,(s+1)}\right)_{GFIC} - \left(\psi_{m,N_x,j}^{x+,(s+1)}\right)_0}{\left(\phi_{k',j'}^{(s)}\right)_{GFIC} - \left(\phi_{k',j'}^{(s)}\right)_0} \quad \text{and} \quad (3.79)$$

$$j_{\psi,(m,k,N_y),(k',j')} = \frac{\Delta\psi_{m,k,N_y}^{y+,(s+1)}}{\Delta\phi_{k',j'}^{(s)}} = \frac{\left(\psi_{m,k,N_y}^{y+,(s+1)}\right)_{GFIC} - \left(\psi_{m,k,N_y}^{y+,(s+1)}\right)_0}{\left(\phi_{k',j'}^{(s)}\right)_{GFIC} - \left(\phi_{k',j'}^{(s)}\right)_0}$$

$$k_{\phi,(k,j),(m,1,j')} = \frac{\Delta\phi_{k,j}^{(s+1)}}{\Delta\psi_{m,1,j'}^{x-,(s+1)}} = \frac{\left(\phi_{k,j}^{(s+1)}\right)_{GFIC} - \left(\phi_{k,j}^{(s+1)}\right)_0}{\left(\psi_{m,1,j'}^{x-,(s+1)}\right)_{GFIC} - \left(\psi_{m,1,j'}^{x-,(s+1)}\right)_0} \quad \text{and} \quad (3.80)$$

$$k_{\phi,(k,j),(m,k',1)} = \frac{\Delta\phi_{k,j}^{(s+1)}}{\Delta\psi_{m,k',1}^{y-,(s+1)}} = \frac{\left(\phi_{k,j}^{(s+1)}\right)_{GFIC} - \left(\phi_{k,j}^{(s+1)}\right)_0}{\left(\psi_{m,k',1}^{y-,(s+1)}\right)_{GFIC} - \left(\psi_{m,k',1}^{y-,(s+1)}\right)_0}$$

$$k_{\psi,(m,N_x,j),(m,1,j')} = \frac{\Delta\psi_{m,N_x,j}^{x+,(s+1)}}{\Delta\psi_{m,1,j'}^{x-,(s+1)}} = \frac{\left(\psi_{m,N_x,j}^{x+,(s+1)}\right)_{GFIC} - \left(\psi_{m,N_x,j}^{x+,(s+1)}\right)_0}{\left(\psi_{m,1,j'}^{x-,(s+1)}\right)_{GFIC} - \left(\psi_{m,1,j'}^{x-,(s+1)}\right)_0}, \quad (3.81)$$

$$k_{\psi,(m,k,N_y),(m,1,j')} = \frac{\Delta\psi_{m,k,N_y}^{y+,(s+1)}}{\Delta\psi_{m,1,j'}^{x-,(s+1)}} = \frac{\left(\psi_{m,k,N_y}^{y+,(s+1)}\right)_{GFIC} - \left(\psi_{m,k,N_y}^{y+,(s+1)}\right)_0}{\left(\psi_{m,1,j'}^{x-,(s+1)}\right)_{GFIC} - \left(\psi_{m,1,j'}^{x-,(s+1)}\right)_0},$$

$$k_{\psi,(m,N_x,j),(m,k',1)} = \frac{\Delta\psi_{m,N_x,j}^{x+,(s+1)}}{\Delta\psi_{m,k',1}^{y-,(s+1)}} = \frac{\left(\psi_{m,N_x,j}^{x+,(s+1)}\right)_{GFIC} - \left(\psi_{m,N_x,j}^{x+,(s+1)}\right)_0}{\left(\psi_{m,k',1}^{y-,(s+1)}\right)_{GFIC} - \left(\psi_{m,k',1}^{y-,(s+1)}\right)_0}, \quad \text{and}$$

$$k_{\psi,(m,k,N_y),(m,k',1)} = \frac{\Delta\psi_{m,k,N_y}^{y+,(s+1)}}{\Delta\psi_{m,k',1}^{y-,(s+1)}} = \frac{\left(\psi_{m,k,N_y}^{y+,(s+1)}\right)_{GFIC} - \left(\psi_{m,k,N_y}^{y+,(s+1)}\right)_0}{\left(\psi_{m,k',1}^{y-,(s+1)}\right)_{GFIC} - \left(\psi_{m,k',1}^{y-,(s+1)}\right)_0}$$

In these equations,  $\Delta$  indicates the change in the corresponding quantity. The subscripts *GFIC* and “0” generically refer to values from two different states, often referred to as *initial* and *final*. Thus, by imposing appropriate “0” and *GFIC* states, the elements of the *ITMM* matrices are constructed using the solution obtained with a generic *SI* sweep performed locally in the sub-domain. For the “0” state, we prescribe a scenario in which the sub-domain has no external source, a zero initial guess, and vacuum boundary conditions. Under this specification, all terms with the subscript “0” are clearly equal to zero, thus allowing their elimination from the equations. Moving to the *GFIC* state, employing the Green’s Function strategy we may only change from the “0” state the single value which is in the denominator given the cell  $(k', j')$  (and potentially angle  $m$ ) corresponding to the progression within the outer sweep. Thus, for construction of  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ , we impose  $\left(\phi_{k',j'}^{(s)}\right)_{GFIC} = 1$  for the cell  $(k', j')$ , leaving all other values the same as the “0” state. Under these “0” and *GFIC* state specifications, Eqs. (3.78) and (3.79) become,

$$j_{\phi,(k,j),(k',j')} = \left(\phi_{k,j}^{(s+1)}\right)_{GFIC} \quad (3.82)$$

$$j_{\psi,(m,N_x,j),(k',j')} = \left(\psi_{m,N_x,j}^{x+,(s+1)}\right)_{GFIC} \quad \text{and} \quad (3.83)$$

$$j_{\psi,(m,k,N_y),(k',j')} = \left(\psi_{m,k,N_y}^{y+,(s+1)}\right)_{GFIC} .$$

Similarly, to construct  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$ , we impose  $\left(\psi_{m,1,j'}^{x-,(s+1)}\right)_{GFIC} = 1$  or  $\left(\psi_{m,k',1}^{y-,(s+1)}\right)_{GFIC} = 1$  for an incoming sub-domain boundary face at direction  $m$ , leaving all other values the same as the “0” state. The “0” and *GFIC* state specifications thus simplify Eqs. (3.80) and (3.81) to,

$$k_{\phi,(k,j),(m,1,j')} = \left(\phi_{k,j}^{(s+1)}\right)_{GFIC} \quad \text{and} \quad k_{\phi,(k,j),(m,k',1)} = \left(\phi_{k,j}^{(s+1)}\right)_{GFIC} \quad (3.84)$$

and

$$\begin{aligned}
k_{\psi,(m,N_x,j),(m,1,j')} &= \left( \psi_{m,N_x,j}^{x+,(s+1)} \right)_{GFIC} , \\
k_{\psi,(m,k,N_y),(m,1,j')} &= \left( \psi_{m,k,N_y}^{y+,(s+1)} \right)_{GFIC} , \\
k_{\psi,(m,N_x,j),(m,k',1)} &= \left( \psi_{m,N_x,j}^{x+,(s+1)} \right)_{GFIC} , \quad \text{and} \\
k_{\psi,(m,k,N_y),(m,k',1)} &= \left( \psi_{m,k,N_y}^{y+,(s+1)} \right)_{GFIC}
\end{aligned} \tag{3.85}$$

Equations (3.82) - (3.85) prescribe the elements of the *ITMM* matrices constructed using *GFIC*. In these equations, note that the left-hand-side contains the indices of cell  $(k', j')$ , while the right-hand-side does not. These indices correspond to the cell in which the lagged cell-averaged scalar flux or incoming angular flux equal to 1 was imposed. The remaining terms in these four equations with the subscript *GFIC* refer to the value in cell  $(k, j)$  after the execution of one local *SI* iteration given the imposed flux in cell  $(k', j')$ .

In general, the algorithm for *ITMM* construction via *GFIC* is simple; create a “0” state for the problem, effectively zero values for all relevant quantities, then impose a value of 1 to the quantity whose effect on other quantities is to be calculated, then execute one local *SI* iteration to obtain these responses.

More specifically, to construct one column of the matrices  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ , specify the lagged cell-averaged scalar flux in cell  $(k', j')$  to be equal to 1 with all other lagged cell-averaged scalar fluxes, external sources, and sub-domain boundary conditions set to zero. Then execute one *SI* iteration over the sub-domain. Upon completion of this iteration, the resulting cell-averaged scalar flux in cell  $(k, j)$  is the entry  $j_{\phi,(k,j),(k',j')}$ , and the resulting outgoing sub-domain boundary flux for direction  $m$ , cell  $(N_x, j)$  or  $(k, N_y)$ , is the entry  $j_{\psi,(m,N_x,j),(k',j')}$  or  $j_{\psi,(m,k,N_y),(k',j')}$  respectively. Thus, executing this procedure once constructs one column of both  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ . Repeating this procedure so that indices  $k'$  and  $j'$  span the sub-domain constructs all columns of these two matrices. Note that the *SI* iteration may be executed over all angles for a given cell  $(k', j')$  before advancing the outer sweep, thus producing the *GFIC* cell-averaged scalar fluxes directly. Alternatively,  $(k', j')$  may span the sub-domain for a single angle, thus producing *GFIC* cell-averaged angular fluxes. The *GFIC* cell-averaged scalar fluxes in this scenario are calculated as a running summation of the *GFIC* cell-averaged angular fluxes multiplied by their respective quadrature weights.

Analogously, to construct one column of the matrices  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$ , specify the incoming sub-domain boundary angular flux for direction  $m$  in cell  $(1, j')$  or  $(k', 1)$  (for an angle in the first quadrant) to be equal to one with all other sub-domain boundary conditions, lagged cell-averaged scalar fluxes, and external sources equal to zero. Then, execute one local mesh sweep used in the *SI* solution for direction  $m$ . Upon completion of this sweep, the resulting cell-averaged scalar flux in a cell  $(k, j)$  is the entry  $k_{\phi, (k, j), (m, 1, j')}$  or  $k_{\phi, (k, j), (m, k', 1)}$  respectively. If the incoming boundary angular flux was specified at the  $x$  sub-domain boundary, the outgoing sub-domain boundary flux for direction  $m$ , cell  $(N_x, j)$  or  $(k, N_y)$  is the entry  $k_{\psi, (m, N_x, j), (m, 1, j')}$  or  $k_{\psi, (m, k, N_y), (m, 1, j')}$  respectively. If the incoming boundary angular was specified at the  $y$  sub-domain boundary, the outgoing sub-domain boundary flux for direction  $m$ , cell  $(N_x, j)$  or  $(k, N_y)$  is the entry  $k_{\psi, (m, N_x, j), (m, k', 1)}$  or  $k_{\psi, (m, k, N_y), (m, k', 1)}$  respectively. Executing this procedure once constructs one column of both  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$ . Repeating this procedure so that the indices  $k'$ ,  $j'$ , and  $m$  span all incoming sub-domain boundary angular fluxes for all directions constructs all columns of these two matrices. Note that the indices stated in this paragraph were for sub-domain boundary incoming angular fluxes in the first quadrant. This concludes the description of our new algorithm for constructing the *ITMM* matrices using a standard, but local, *SI* mesh sweep via the *GFIC* algorithm.

### 3.2.5 Computational Cost of *ITMM* Matrix Construction Algorithms

The *GFIC* algorithm was developed to enable construction of the *ITMM* matrices in unstructured meshes using the pre-existing capability of *SI* in a typical  $S_N$  code. Recall that with the differential mesh sweep, the inner sweep only swept over cells downstream from cell  $(k', j')$ . In the mesh sweep with *GFIC*, the same is conceptually true. However, doing so on unstructured grids would require modifying the *SI* mesh sweep algorithm to sweep over only a portion of the associated sub-domain, thus diminishing the simplicity of implementation that was our purpose for the development of the *GFIC* algorithm. To avoid this added complication, the *SI* iteration may be executed over the entire sub-domain, which clearly does not change the produced elements, as the solution in upstream cells is zero. We refer to the practice of only sweeping over downstream cells as the “ideal sweep” and sweeping over the whole sub-domain as the “full sweep”.

The simplicity of implementation on unstructured grids with the full sweep comes at the cost of increased construction time of the *ITMM* matrices compared to the ideal sweep. We therefore conduct theoretical analysis to quantify the difference in cost of constructing the four *ITMM* matrices between the full sweep and ideal sweep. Regardless of the sweep type used, the computational work required for each individual cell is the same. Additionally, the required number of operations for the *GFIC* algorithm scales linearly with the number of discrete ordinates. For these reasons, we calculate the number of cells that are solved using the ideal or full sweep to perform both outer sweeps for a single direction. This provides expressions that are proportional to the cost of construction for both sweep types, with the multiplicative factor remaining the same for both, namely the *SI* grind time.

To begin, consider a 2-D,  $N \times N$  cell sub-domain and a direction  $m$  in any quadrant. To perform one outer sweep for a single direction, the number of cells that must be solved for the full sweep case is trivial, as the inner sweep spans the whole sub-domain independent of progress through the outer sweep. For the full sweep, the number of cells that must be solved to execute both single outer sweeps for one direction in 2-D geometry is,

$$T_{full,2D} = N^4 + 2N^3. \quad (3.86)$$

This value is proportional to the construction time required to construct the four *ITMM* matrices with the *GFIC* algorithm using a full sweep. To derive this expression, consider that  $N^2$  cells must be solved in the inner sweep at every step of the outer sweep. To construct  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ , the outer sweep must span all cells in the sub-domain, thus requiring the inner sweep to be executed  $N^2$  times for a total of  $N^4$  cells to be solved. To construct  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$ , the outer sweep spans all incoming sub-domain boundary faces, thus requiring the inner sweep to be executed  $2N$  times for a total of  $2N^3$  cells to be solved.

The number of cell solutions required to execute both outer sweeps for a single direction using the ideal sweep is more complicated, as the number of cells spanned by the inner sweep changes based on position within the outer sweep. For the construction of  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$ , one inner sweep for a direction  $m$  in the third quadrant initiated from cell  $(k', j')$  in the outer sweep spans  $k' \times j'$  cells. With the outer sweep for construction of  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$  spanning all cells in the sub-domain, the number of cell solutions required to execute one outer sweep for the construction of

these two matrices is  $\sum_{k'=1}^N k' \sum_{j'=1}^N j' = \left(\frac{N(N+1)}{2}\right)^2$ . For the construction of  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$ , either  $k'$  or  $j'$  is required to be equal to  $N$ , thus resulting in the number of cells to be spanned by the inner sweep to be  $Nj'$  or  $Nk'$  for incoming sub-domain boundary faces on the  $x$  or  $y$  boundary respectively. Summing over the incoming sub-domain boundary faces that are spanned by the outer sweep, the number of cells which must be solved to execute one outer sweep for the construction of  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$  is  $\sum_{k'=1}^N Nk' + \sum_{j'=1}^N Nj' = N^2(N+1)$ .

The number of cells that are solved in one outer sweep for a single direction using the ideal sweep is therefore,

$$T_{ideal,2D} = \frac{N^2(N+1)(N+5)}{4}. \quad (3.87)$$

Equations (3.86) and (3.87) are the required numbers of cell solutions to execute one outer sweep of the *GFIC* algorithm for a single direction using a full or ideal sweep respectively. These equations are proportional to the total time required to construct the four *ITMM* matrices using either sweep mechanism with the proportionality coefficient in both formulations equal to the standard grind time. They may thus be used as a surrogate for total construction time when predicting the penalty of increased construction time associated with the full sweep. These theoretical predictions are compared to computational observation in Sec. 4.9.

For a 3-D,  $N \times N \times N$  sub-domain, analogous calculations yield the following 3-D expressions.

$$T_{full,3D} = N^6 + 3N^5 \quad (3.88)$$

$$T_{ideal,3D} = \frac{N^3(N+1)^2(N+7)}{8} \quad (3.89)$$

### 3.3 Summary of Theoretical Analysis

Our theoretical analysis consists of two main elements: Fourier analysis of *P-PI-SI* and *P-IP-SI*, and construction algorithms for the *ITMM* matrices. The Fourier analysis of *P-PI-SI* and *P-*

*IP-SI* derived expressions for the scaling of the iterative error eigenmodes due to a single iteration of each method as a function of the frequency variables in Fourier space. These expressions, Eqs. (3.33) and (3.47) will subsequently be used to verify the implementation of these acceleration techniques in a 2-D, Cartesian grid test code, HAT-2C, as well as inform development of asynchronous hybrid iterative methods. The Fourier analysis of *P-IP-SI* is largely used, not in support of *P-IP-SI*, but to confirm conjectures made about the underlying mechanisms of our hybrid approach. The primary conjecture which we use this Fourier analysis to successfully confirm is that *P-PI-SI* is effective due to *SI*'s ability to resolve long-distance streaming and *PBJ-ITMM*'s ability to resolve local collision.

We have also presented algorithms for the construction of the matrices required for *ITMM*'s implementation. The two algorithms discussed were the previously reported *DMS* and the newly developed *GFIC* algorithms. We developed the latter of these algorithms motivated by its potential for direct extendibility to unstructured grids. This algorithm selectively imposes unit flux values in cells or on incoming sub-domain boundaries (a discrete Green's Function prescription) that renders the *SI* sweep solution equivalent to the *DMS* solution, thus eliminating the need to implement a new kernel calculation module for matrix construction in an existing transport code that seeks to employ *PBJ-ITMM* as a solution option.

## Chapter 4:

# Numerical Experimentation with Hybrid Methods in 2-D Cartesian Geometry with Serial Operation

To continue the development of our hybrid approach as well as preliminarily test *GFIC*, we conduct a set of computational experiments in serial execution on 2-D Cartesian grids. At this early development stage, this simplified configuration is preferred for its practicality and to expedite the development process. With numerous iterative methods being developed, implementation, debugging, and testing progress faster on Cartesian grids than on unstructured. Additionally, it is advantageous to verify *GFIC* on Cartesian grids against *DMS* as a reference before implementing it on unstructured grids. For our studies, 2-D geometry is preferable over 3-D as it results in greatly reduced execution times, allowing large amounts of data to be collected expeditiously. Finally, serial operation is used because the iterative convergence rates of our methods are unaffected by whether the operations are performed in series or in parallel. With methods in the development stage, serial operation allows expedited development as well as execution on a personal desktop computer as opposed to an HPC, which can be a difficult to acquire resource for untested methods and may also impose delays due to the large number of concurrent users. With this simplified configuration, we develop and test our methods recognizing that they are intended to be extendible to the target application of solving the  $S_N$  equations on massively parallel computing platforms on unstructured grids. The main consideration for this is that, when decreeing operations as being parallelizable, we must assume that spatially parallel sweeps are not available, as our goal is to provide as alternative to such sweep algorithms on unstructured grids. With regards to this consideration, in this chapter we frequently discuss methods' degrees of parallelism. For the entirety of this chapter, this degree of parallelism refers to the intuitively expected degree of parallelism on unstructured grids and is not based on a thorough consideration of parallelization options.

Our computational experiments utilize a newly developed 2-D, Cartesian grid transport test code that uses Fortran 95 to implement the *AHOT-N0* spatial discretization for solution of the discrete ordinates equations, referred to as the *Hybrid Spatial Domain Decomposition (HSDD)*

*AHOT-N0* Transport Solver in 2-D Cartesian Geometries (HAT-2C) code. The iterative solution strategies implemented in HAT-2C comprise five iterative method options: *SI*, *PBJ-ITMM*, *IPBJ*, *P-PI-SI*, and *P-IP-SI*. The *SI* acceleration schemes implemented include *DSA*, *AP*, *NDA*, and *pNDA*. The matrix solution associated with *PBJ-ITMM* is performed using the LaPack (Linear Algebra PACKage) [63] routines, which compute and store the *LU* factorization of  $\mathbf{I} - \mathbf{J}_\phi$  after construction, then directly perform the *ITMM* matrix equation operations in each iteration. The low-order matrix solutions required by *DSA*, *AP*, *pNDA*, and *NDA* are performed in each iteration using DLAP [64], a package for the solution of sparse matrices. The associated matrices are solved by DLAP using the *BiCGSTAB* method with *ILU* factorization. [65]

One of the features we have implemented in HAT-2C for our research is the *Hybrid Spatial Domain Decomposition (HSDD)*. The *HSDD* decomposes the spatial domain into sub-regions (not to be confused with the sub-domains associated with *PBJ*), termed “zones”. The incoming angular fluxes at the zone interfaces are lagged by an iteration, exchanging these values with adjacent neighboring zones between iterations. With all zones decoupled from each other, any of the five iterative methods implemented in HAT-2C may be specified in any given zone to apply the iterative solution. The iterative method specified for any given zone is independent of the method specified for all other zones, thus allowing zones to use different iterative methods than others. The *HSDD* was implemented to allow implicit support of the additional iterative methods, *AH-PI-SI* (asynchronous *PBJ-ITMM* / *SI* hybrid method), *AH-IP-SI* (asynchronous *IPBJ* / *SI* hybrid method), and *AH-PI-IP* (asynchronous *PBJ-ITMM* / *IPBJ* hybrid method). These methods execute only their primary method in zones containing optically thick cells and only their secondary method in zones containing optically thin cells. The development of these methods emerged from initial computational experiments described below, and the motivation for these methods’ creation will be thoroughly explained as part of that presentation. The results of our numerical experiments conducted using HAT-2C are organized as follows.

Firstly, in Sec. 4.1 we perform verification of previously developed iterative methods for which results to compare against are available in the literature. This verification is conducted by reproducing tables of the number of required iterations to achieve a set stopping criterion (loosely referred to as convergence criterion in previous chapters of this dissertation) or spectral radius trends previously reported in the literature.

We then use our HAT-2C code in Sec. 4.2 to computationally verify the Fourier analysis for  $P$ - $PI$ - $SI$  and  $P$ - $IP$ - $SI$ . With both the code and our Fourier analyses verified, we discuss the results of these Fourier analyses. This theoretical analysis demonstrates three primary concepts: 1.  $IPBJ$  is likely less effective than  $PBJ$ - $ITMM$  as the primary method in our proposed hybrid approach due to the lagging of the scattering source; 2. the conjecture that the underlying mechanism of an effective hybrid method of  $PBJ$ -type and sweep-based methods is the accurate modeling of long-distance streaming by  $SI$  paired with the resolution of local collisions by the  $PBJ$  method; and 3. that executing both methods over the entire spatial domain is likely unnecessary.

Section 4.3 briefly presents the spectral radius of  $PBJ$ - $ITMM$  and  $IPBJ$  as functions of fundamental transport phenomena governed by the nuclear properties  $\Sigma_s$  and  $\Sigma_a$ . These trends are used to conceptually explain  $PBJ$ - $ITMM$ 's lack of iterative robustness with respect to scattering ratio despite the resolution of the scattering source, as well as subsequent observations of  $PBJ$ - $ITMM$ 's iterative performance in highly scattering media.

In Sec. 4.4 We perform experiments that compare, in a homogeneous medium, the iterative performance of our  $SI$  preconditioning methods to that of the individual iterative methods of which they are comprised. The results demonstrate that one of the two iterative methods used is largely responsible for the convergence of a given cell's solution with little assistance from the other, thus indicating that a more advantageous approach is the *asynchronous hybrid* methods. This approach is tested along with the preconditioning approach on a heterogeneous medium in Sec. 4.5.

With the *asynchronous hybrid* approach developed, in Sec. 4.6 we conduct a parametric study using a heterogeneous problem with regions alternating between optically thin and thick cells. This study is conducted to assess the iterative performance of our hybrid methods in comparison to each other as well as against traditional  $DSA$ ,  $AP$ , and  $pNDA$ , and how each method's performance is impacted by various problem parameters. This parametric study is extended to extremely large scattering ratios in Sec. 4.7. With the gained understanding of how hybrid methods will perform in certain problems, we test our approach on realistic test problems containing void regions in Sec. 4.8.

Finally, in Sec. 4.9 we compare the construction cost of the  $ITMM$  matrices when using the  $GFIC$  algorithm against the  $DMS$ . This comparison includes both the ideal and full sweep variations of the mesh sweep utilized by  $GFIC$ .

## 4.1: HAT-2C Code Verification

We begin the verification process with a comparison of the required number of iterations for *SI* with and without *DSA* to those previously reported in the literature, specifically reference [13]. In the aforementioned work, the author uses *SI* with and without *DSA* to solve the problem depicted in Fig. 4.1 for a variety of material specifications, presenting the number of iterations required. To verify the implementation of *SI* and *DSA* in our HAT-2C code, we repeat some of these tests, verifying that our code produces comparable iteration counts.

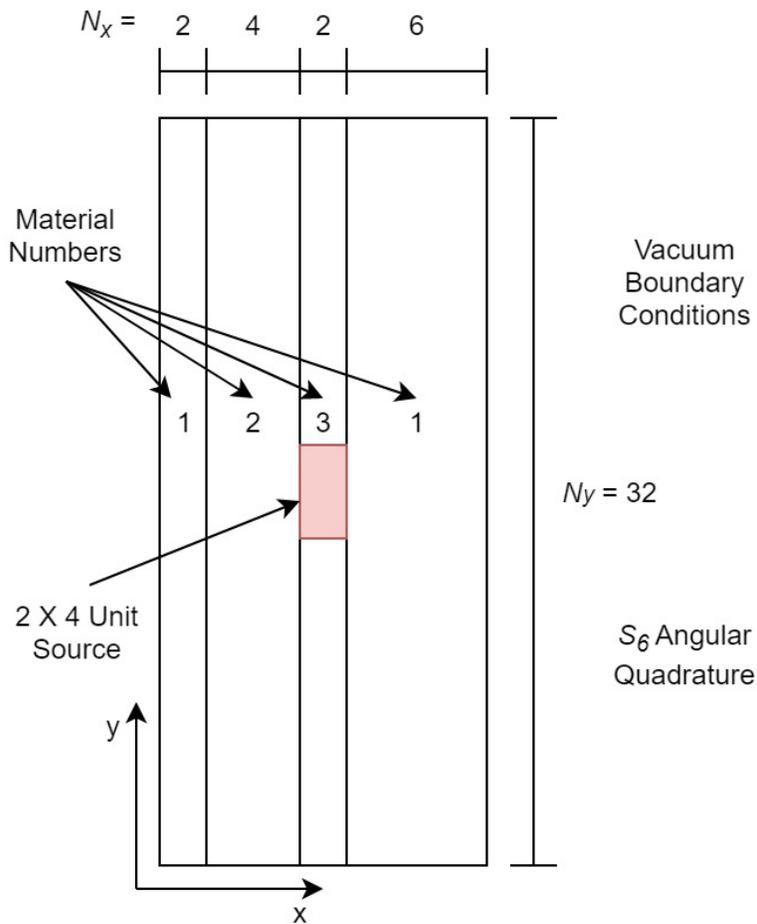


Figure 4.1: Setup of verification problem with varying  $c$  and cell size for *SI* and *DSA* from [13]

The first test we replicate is a homogenous case in which the three materials have identical cross sections. The number of iterations required to achieve a  $10^{-6}$  relative stopping criterion for

this configuration is reported for *SI* and *DSA* for various scattering ratios with all other problem specifications fixed, in Table 4.1.

Table 4.1: Iteration counts required by HAT-2C to converge the problem depicted in Fig. 4.1 for the homogeneous case using *SI* and *DSA* for multiple scattering ratios.  $\Sigma_t \Delta u = 1.0$ ,  $u = x$  or  $y$ , relative stopping criterion:  $\epsilon = 10^{-6}$ .

$c$	<i>SI</i>	<i>DSA</i>
0.5	33	7
0.8	79	8
0.98	319	8

A second test using the homogeneous medium is displayed in Table 4.2, with the number of required iterations shown as a function of cell size.

Table 4.2: Iteration counts required by HAT-2C to converge the problem depicted in Fig. 4.1 for the homogeneous case using *SI* and *DSA* for multiple cell sizes.  $c = 0.98$ , relative stopping criterion:  $\epsilon = 10^{-6}$ .

Cell Size ( <i>mfp</i> )	<i>SI</i>	<i>DSA</i>
0.5	165	9
1.0	319	8
2.0	493	8
4.0	651	8

Finally, we replicate a heterogeneous problem configuration from Fig. 4.1, the cross sections and required iteration counts for which are displayed in Table 4.3.

Table 4.3: Iteration counts required by HAT-2C to converge the problem depicted in Fig. 4.1 for the heterogeneous case using *SI* and *DSA* and the associated cross sections for the three materials.  $\Delta u = 2.0, u = x$  or  $y$ , relative stopping criterion:  $\epsilon = 10^{-5}$ .

$\Sigma_{t,1}$	$\Sigma_{s,1}$	$\Sigma_{t,2}$	$\Sigma_{s,2}$	$\Sigma_{t,3}$	$\Sigma_{s,3}$	<i>SI</i>	<i>DSA</i>
0.330263	0.314419	0.694676	0.634883	0.499122	0.49446	167	7

The required number of iterations for *SI* and *DSA* using our HAT-2C are consistent with those previously reported. [13] These results verify the implementation of these two methods in HAT-2C.

We now verify the implementation of *AP* in HAT-2C, reproducing results from [20]. The test problem used to obtain these results is depicted in Fig. 4.2.

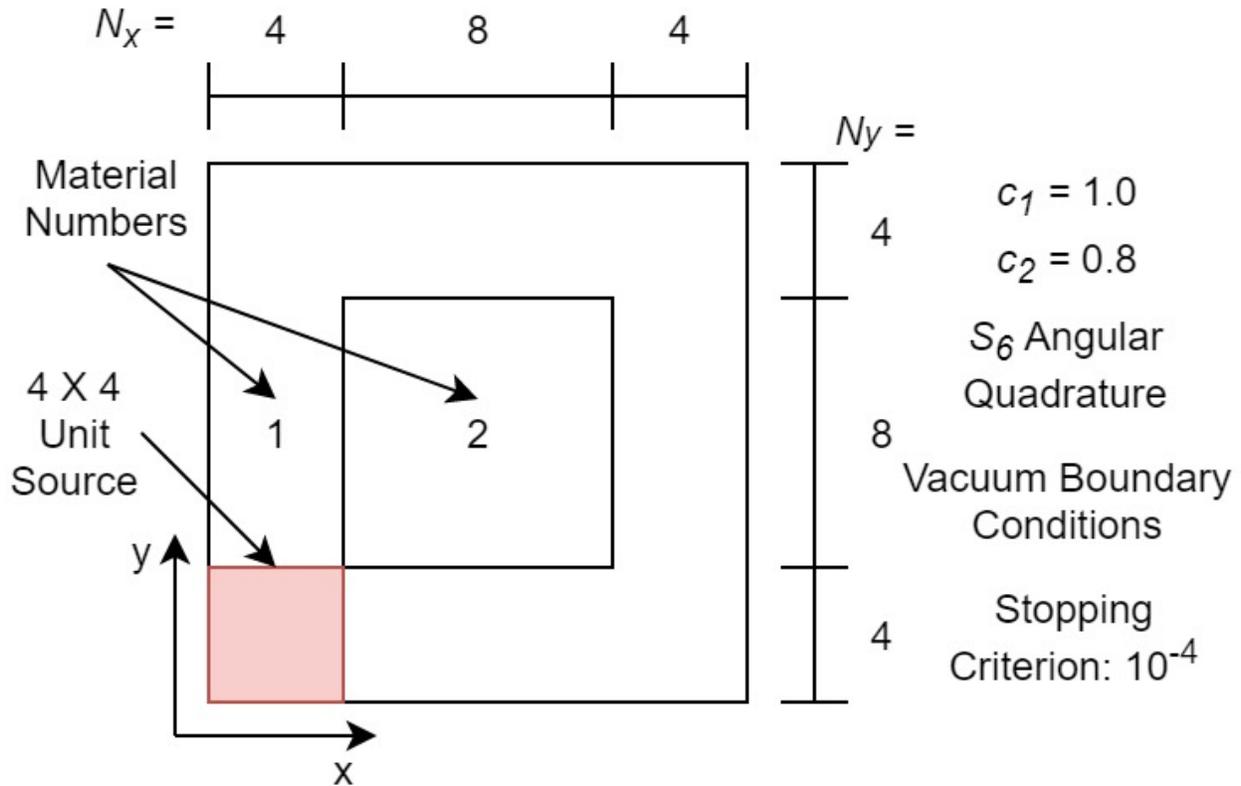


Figure 4.2: Setup of verification problem for *AP* from [20].  $\Delta x = 1.0$ .

The iteration counts required for converging this problem with our HAT-2C transport code using *AP* are displayed in Table 4.4 for multiple values of different problem parameters. These iteration counts are consistent with previously reported data [20], thus verifying our implementation of *AP*.

Table 4.4: Iteration counts required by HAT-2C to converge the problem depicted in Fig. 4.2 using *AP* for multiple values of  $\Sigma_{t,1}$ ,  $\Sigma_{t,2}$ ,  $\Delta y_1$ , and  $\Delta y_2$ . All other problem parameters are fixed as indicated in Fig. 4.2.

			$\Sigma_{t,1}$											
			0.01			0.1			1.0			10.0		
			$\Delta y_1 \rightarrow$	0.01	0.1	1.0	0.01	0.1	1.0	0.01	0.1	1.0	0.01	0.1
$\Sigma_{t,2}$	0.01	$\Delta y_2 \downarrow$												
		0.01	4	5	5	5	7	7	7	14	10	17	36	13
		0.1	5	5	5	7	9	8	12	19	15	18	38	22
	0.1	0.01	4	5	5	4	8	7	6	11	6	13	20	6
		0.1	6	7	6	6	8	7	9	12	8	19	19	10
		1.0	7	10	7	8	7	7	8	9	9	10	11	10
	1.0	0.01	6	7	6	7	8	7	6	10	5	11	10	5
		0.1	12	16	10	9	12	9	9	11	5	12	7	6
		1.0	10	11	11	8	8	9	6	6	5	6	6	5
	10.0	0.01	12	14	11	8	13	9	7	11	5	13	6	5
		0.1	10	13	10	7	10	9	8	11	6	8	6	5
		1.0	8	10	9	8	9	9	8	9	6	7	7	6

*NDA* and *pNDA* results were not available for us to compare against for the dimensionality and spatial discretization method employed by HAT-2C. However, for all results presented in this chapter, the scalar fluxes were compared across all methods, ensuring that they were equivalent to within the stopping criterion. This process helps to verify our *NDA* and *pNDA* implementations, as the closure between the high-order and low-order problems in *NDA* methods means that an error in implementation would likely result in an erroneous solution that differs from the solution obtained by other, verified methods in excess of the stopping criterion. Additionally, the convergence rates observed throughout for these methods were consistent with expectation.

Next, we verify the implementation of *PBJ-ITMM* and *IPBJ* in our HAT-2C code by measuring spectral radius trends and comparing to the theoretical trends reported in the literature. [4] The spectral radius is estimated in HAT-2C each iteration as the relative change in the  $L_2$  norm

of the iterative change in the cell-averaged scalar flux between two consecutive iterations, a ratio that is expected to converge to the spectral radius as the iterations progress towards convergence. We plot the converged spectral radius estimates from our code for *PBJ-ITMM* and *IPBJ* versus cell size for a homogeneous problem with multiple scattering ratios and problem sizes in Figs. 4.3 and 4.4.

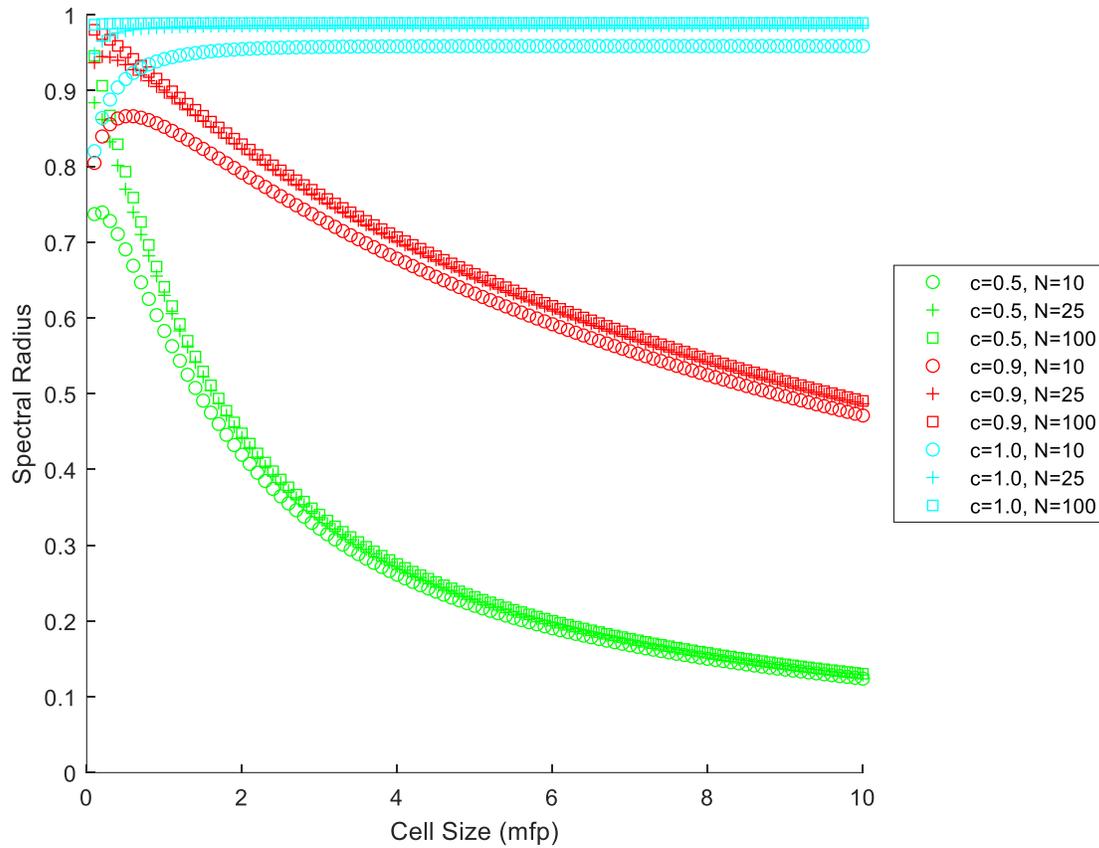


Figure 4.3: Computationally estimated spectral radius produced by HAT-2C for *PBJ-ITMM* versus cell size for multiple scattering ratios of a problem with an  $N \times N$  mesh.

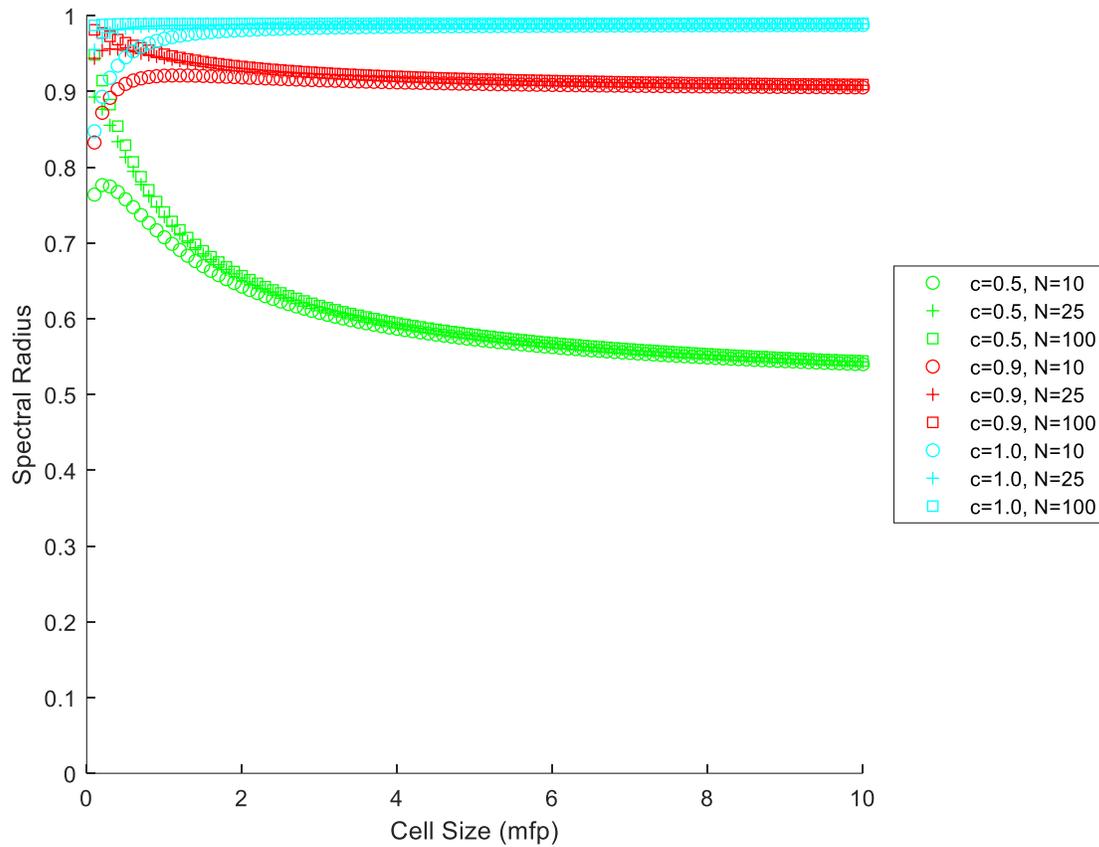


Figure 4.4: Computationally estimated spectral radius produced by HAT-2C for *IPBJ* versus cell size for multiple scattering ratios of a problem with an  $N \times N$  mesh.

The spectral radius trends in Figs. 4.3 and 4.4 verify the implementation of *PBJ-ITMM* and *IPBJ* in HAT-2C, with the computational spectral radius estimates tending towards the theoretically predicted trends previously reported in the literature [4] for the infinite medium as the mesh size increases. In addition to verifying the implementation of these methods in HAT-2C, the observed spectral radius trends provide a visualization of the aforementioned lack of iterative robustness of *PBJ* methods in optically thin cells, as the spectral radius tends towards unity in thin cells for large meshes. This is the feature that our investigation of hybrid methods primarily seeks to address.

For all subsequently presented results, multiple iterative methods are used to solve the same physical problem. Additional verification was performed through testing the scalar flux profiles produced by different methods for the same problem to ensure they are equivalent to within the relative stopping criterion. From the results provided in this section and the comparison to their

respective counterparts in the literature, we conclude that HAT-2C is verified and that results produced from it are trustworthy.

## 4.2: *P-PI-SI* & *P-IP-SI* Fourier Analysis Results

With the capabilities of our HAT-2C code verified as well as a frame of reference of *IPBJ* and *PBJ-ITMM*'s iterative performance provided for comparison, we examine the performance of *P-PI-SI* and *P-IP-SI*. The theoretical results for the Fourier analyses of these methods were obtained using Matlab. [66] We expect the eigenvalues for *P-PI-SI* and *P-IP-SI* to be only real and that the imaginary components of Eqs. (3.33) and (3.47) all sum to zero over the  $M$  angles. Note that this was confirmed computationally by testing the imaginary component for all subsequently displayed cases to ensure that they are zero across all values of the Fourier variables. With this confirmed, the eigenvalues are shown to be real and we henceforth refer to the eigenvalues based on this numerically established, and expected, feature.

We plot the magnitude of the eigenvalues which govern the iterative performance of *P-PI-SI* and *P-IP-SI*, Eqs. (3.33) and (3.47), respectively, versus  $\lambda_x$  and  $\lambda_y$  in Figs. 4.5 - 4.15 for multiple scattering ratios and cell sizes using an  $S_6$  angular quadrature .

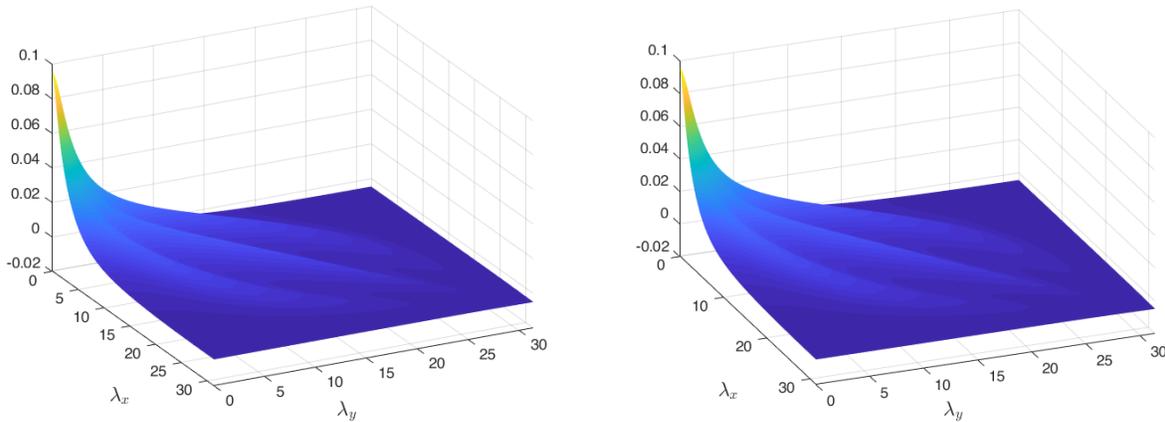


Figure 4.5: *P-PI-SI* (left) and *P-IP-SI* (right) eigenvalues:  $\Sigma_t = 0.1, c = 0.1, \Delta x = 1, \Delta y = 1$

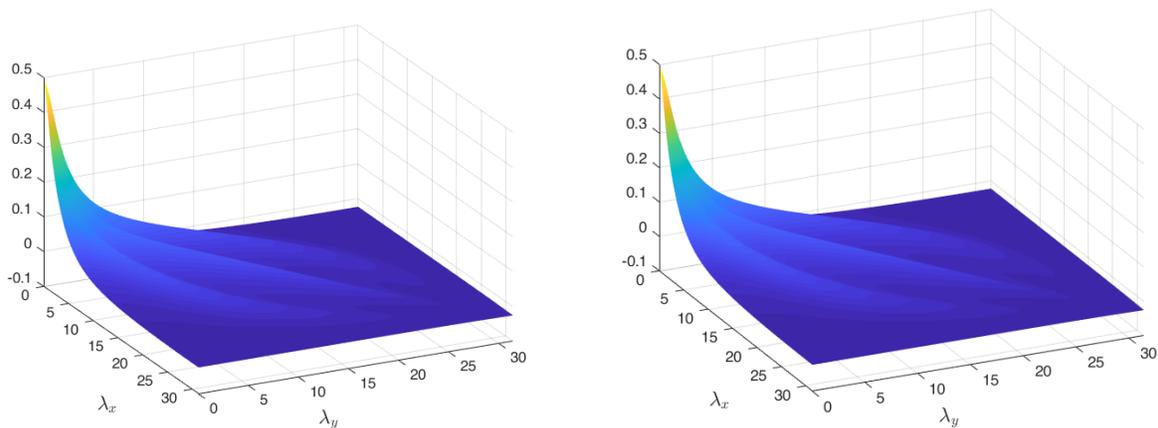


Figure 4.6: *P-PI-SI* (left) and *P-IP-SI* (right) eigenvalues:  $\Sigma_t = 0.1, c = 0.5, \Delta x = 1, \Delta y = 1$

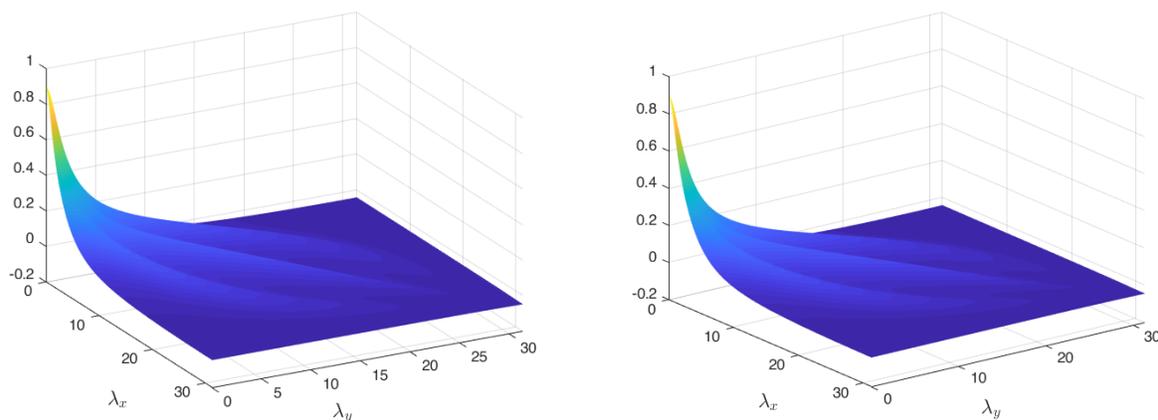


Figure 4.7: *P-PI-SI* (left) and *P-IP-SI* (right) eigenvalues:  $\Sigma_t = 0.1, c = 0.9, \Delta x = 1, \Delta y = 1$

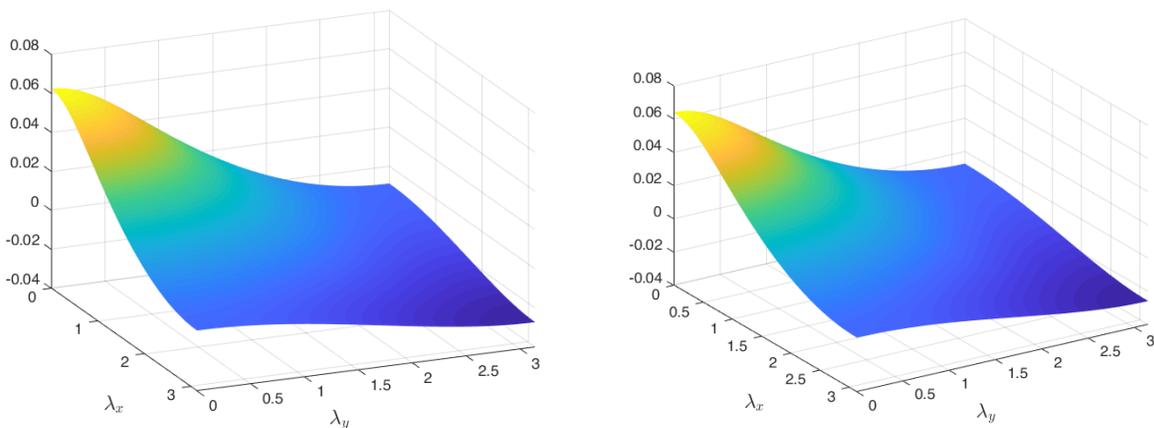


Figure 4.8: *P-PI-SI* (left) and *P-IP-SI* (right) eigenvalues:  $\Sigma_t = 1, c = 0.1, \Delta x = 1, \Delta y = 1$

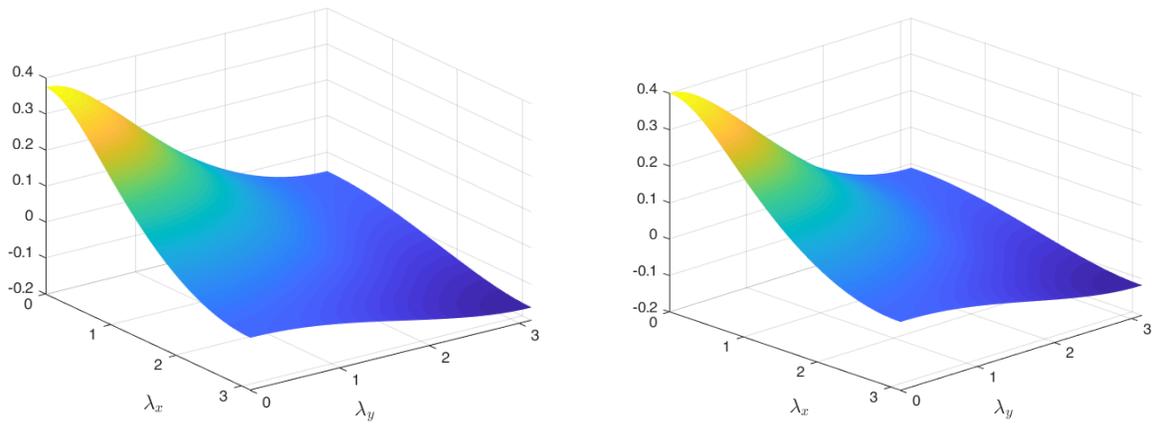


Figure 4.9: *P-PI-SI* (left) and *P-IP-SI* (right) eigenvalues:  $\Sigma_t = 1, c = 0.5, \Delta x = 1, \Delta y = 1$

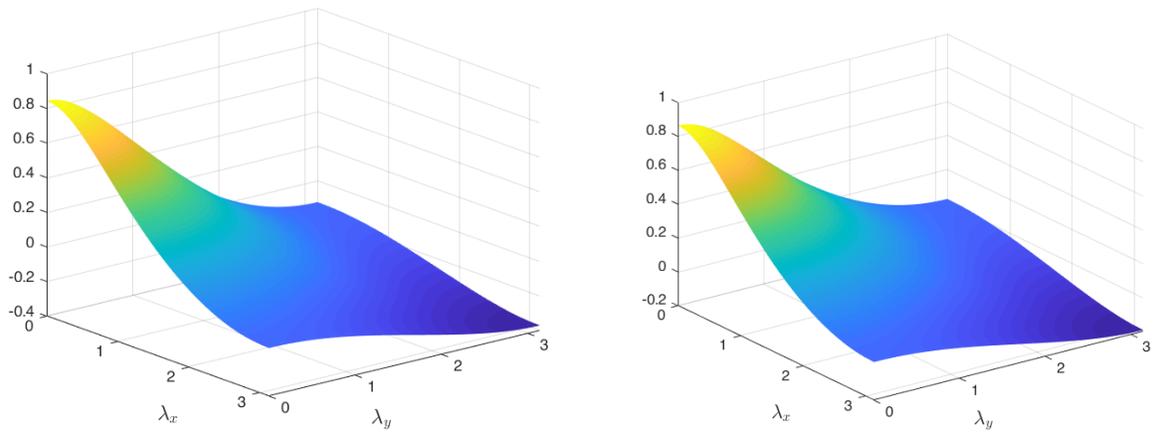


Figure 4.10: *P-PI-SI* (left) and *P-IP-SI* (right) eigenvalues:  $\Sigma_t = 1, c = 0.9, \Delta x = 1, \Delta y = 1$

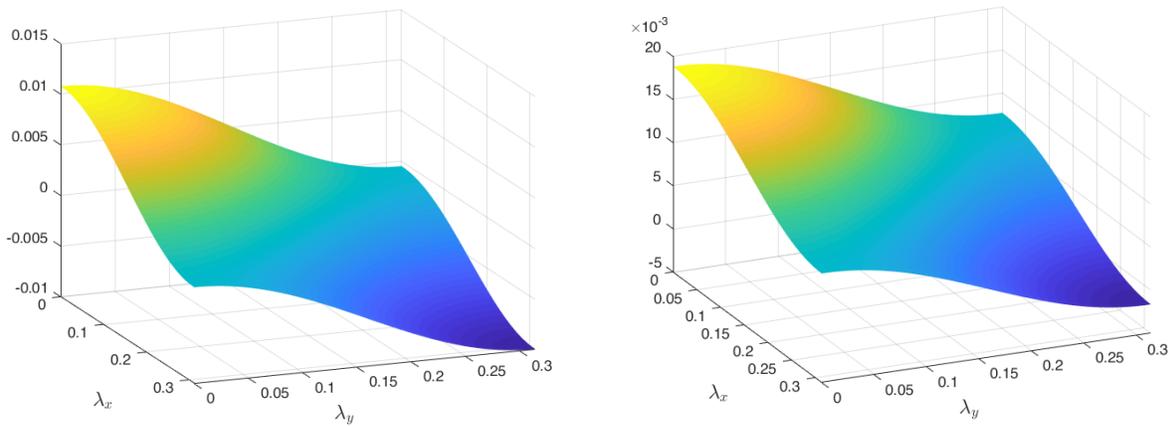


Figure 4.11: *P-PI-SI* (left) and *P-IP-SI* (right) eigenvalues:  $\Sigma_t = 10, c = 0.1, \Delta x = 1, \Delta y = 1$

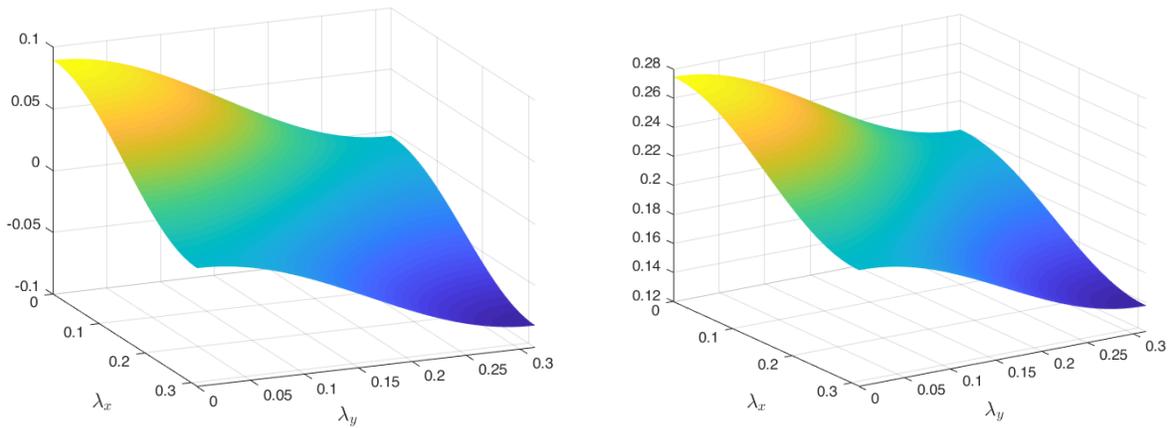


Figure 4.12: *P-PI-SI* (left) and *P-IP-SI* (right) eigenvalues:  $\Sigma_t = 10, c = 0.5, \Delta x = 1, \Delta y = 1$

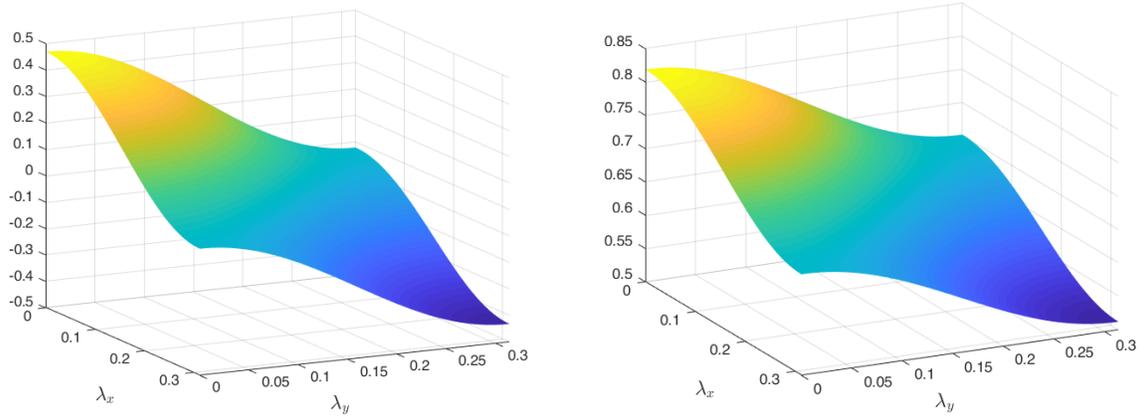


Figure 4.13: *P-PI-SI* (left) and *P-IP-SI* (right) eigenvalues:  $\Sigma_t = 10, c = 0.9, \Delta x = 1, \Delta y = 1$

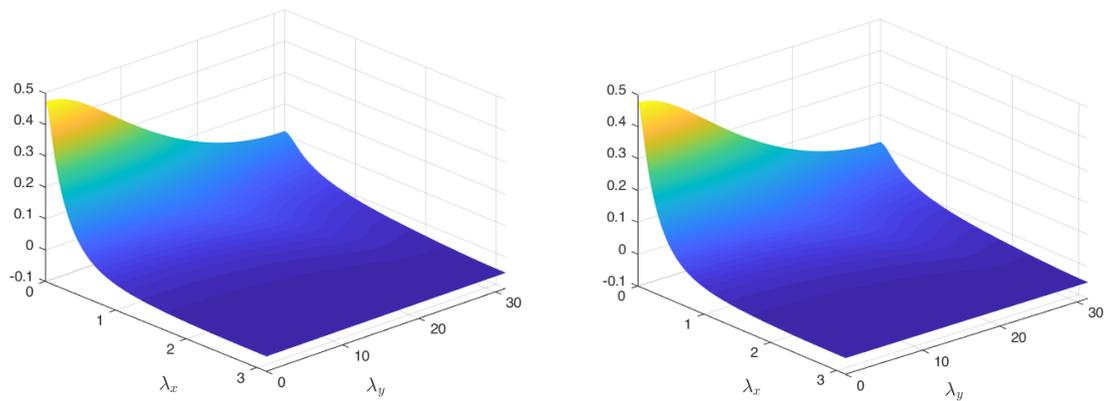


Figure 4.14: *P-PI-SI* (left) and *P-IP-SI* (right) eigenvalues:  $\Sigma_t = 1, c = 0.5, \Delta x = 1, \Delta y = 0.1$

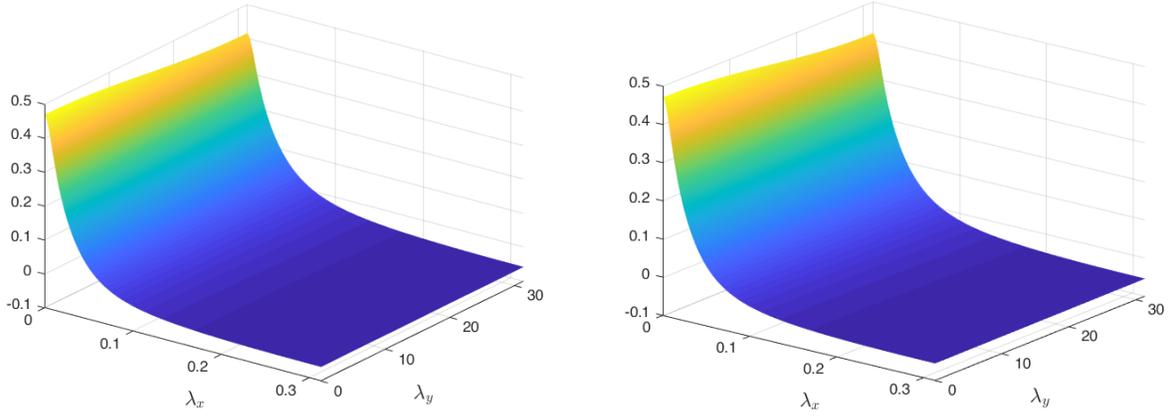


Figure 4.15:  $P$ - $PI$ - $SI$  (left) and  $P$ - $IP$ - $SI$  (right) eigenvalues:  $\Sigma_t = 1, c = 0.5, \Delta x = 10, \Delta y = 0.1$

The primary observation from these eigenvalue plots is that the flat mode ( $\lambda_x = \lambda_y = 0$ ) is the slowest converging mode for both  $P$ - $PI$ - $SI$  and  $P$ - $IP$ - $SI$ . This is clearly observed as the eigenvalue magnitudes are maximized at the Fourier space's origin for all cell sizes and scattering ratios displayed, noting that this peak is not unique due to the periodic nature of Eqs. (3.33) and (3.47). Since the trigonometric functions in these equations contain the cell optical thickness in their arguments, the period of these functions with respect to the Fourier variables becomes smaller with increasing cell optical size; a trait clearly observed in our results, with the changing axis bounds of the Fourier variables for the half-periods plotted.

Additionally, the last two eigenvalue graphs for each method are for cases with different cell sizes in the two dimensions. Comparing these eigenvalues to their counterparts with square cells, we also conclude that the convergence rate of both methods is determined by the optically thinner dimension. This is apparent, as the eigenvalue at the Fourier origin for these cases compares to the case with both dimensions having optical thicknesses equal to the thinner of the two dimensions in the non-square case.

Concluding that the flat mode is the fundamental mode for  $P$ - $PI$ - $SI$  and  $P$ - $IP$ - $SI$ , we plot the magnitude of this eigenvalue, along with computationally estimated values produced by HAT-2C, versus cell size for selected values of the scattering ratio to observe the effect of  $SI$  preconditioning on the iterative convergence rate of  $PBJ$ - $ITMM$  and  $IPBJ$ .

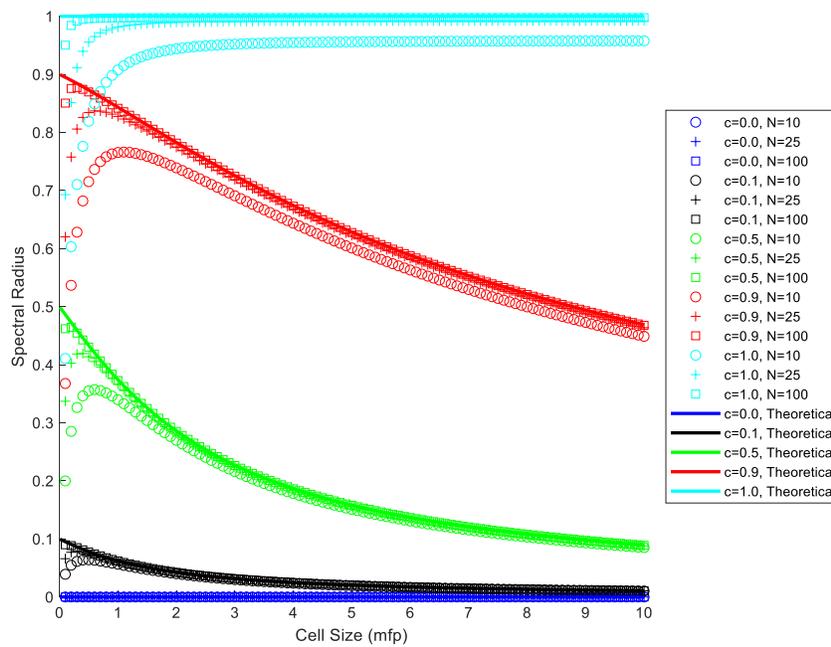


Figure 4.16: Spectral radius of  $P$ - $PI$ - $SI$  estimated with HAT-2C for a homogeneous problem with an  $N \times N$  mesh and the theoretically calculated spectral radius. Spectral radius is graphed versus cell size for multiple scattering ratios using an  $S_6$  angular quadrature.

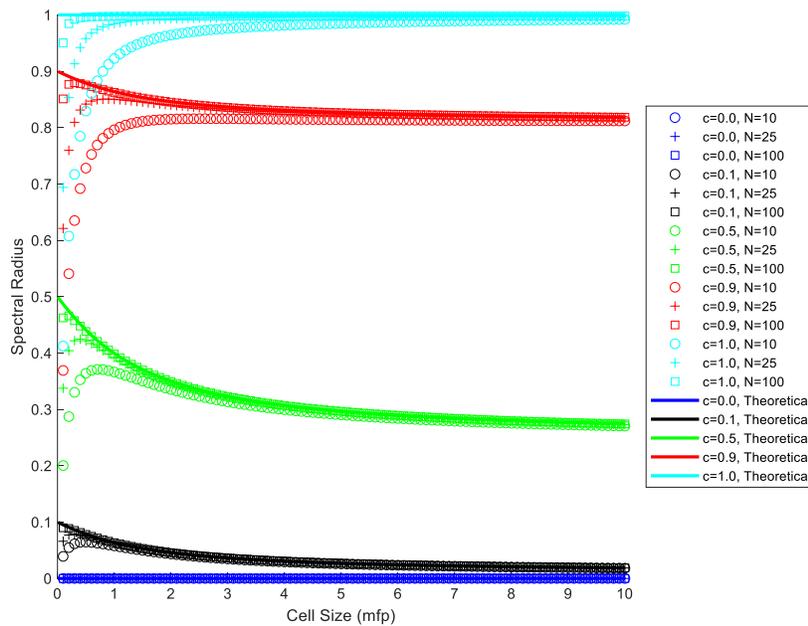


Figure 4.17: Spectral radius of  $P$ - $IP$ - $SI$  estimated with HAT-2C for a homogeneous problem with an  $N \times N$  mesh and the theoretically calculated spectral radius. Spectral radius is graphed versus cell size for multiple scattering ratios using an  $S_6$  angular quadrature.

Analyzing Figs. 4.16 and 4.17, the first thing we note is that the computational results agree with the theoretical predictions. We also see that for all  $c < 1$ , the spectral radius no longer approaches unity as the cell size decreases. Without *SI* preconditioning, as the cell size is decreased, cells become more tightly coupled to distant cells (i.e. two distance cells' solutions become more dependent on each other). Since *PBJ* methods exchange information between sub-domains only in between iterations, significant coupling between unboundedly distant cells (which occurs as the cell size decreases) indicates that the number of iterations required for two significantly coupled cells to become “aware” of a change to the others' solution also grows without bound, thereby reducing robustness. We see that *SI* preconditioning achieved the desired iterative properties. While the spectral radius is still larger in optically thin cells than in uniformly thicker cells, the spectral radius remains bounded below unity for all scattering ratios smaller than 1, thereby recovering the desired robustness for these cases. As stated previously, this is due to *SI*'s synchronous nature which allows all cells to become “aware” of a change in another cell's solution on which its own solution is dependent.

A feature of the *P-PI-SI* and *P-IP-SI* spectra which must be noted is that the reduction of spectral radius in optically thin cells compared to the associated *PBJ* method is greater for low scattering ratios. This is to be expected as the lagged scalar flux in the scattering source of the *SI* equations is known to slow down iterative convergence in problems with scattering ratios close to unity. This unfortunately leads to the smallest benefit from *SI* preconditioning in some of the slower converging problems. Returning to the physical interpretation of the *P-PI-SI* iterative process, *PBJ-ITMM* resolves particle collisions locally within a sub-domain and the *SI* step then allows these particles to stream through the medium until their next collision. In a problem with a high scattering ratio and optically thin cells though, particles can scatter and consequently travel far across the domain many times. The inability of *SI* to resolve collisions over the course of a single iteration renders it ineffective at accounting for this effect.

This physical interpretation of *P-PI-SI* as a method in which *SI* models long-distance streaming while *PBJ-ITMM* models local collision also suggests that *P-IP-SI* will be less effective, and the comparison of the two methods' spectra is the first indication that this is true. With the scattering source lagged, *IPBJ* is unable to model multiple collision events in a single iteration. The underlying mechanism of *P-PI-SI*'s iterative efficiency is therefore not applicable to *P-IP-SI* and it was predicted that the inability to effectively resolve multiple collisions would render the

method ineffective. The first hard evidence of this comes as the  $P$ - $IP$ - $SI$  spectral radius asymptotes in the thick regime towards  $c^2$ , indicating that the method becomes equivalent from the standpoint of convergence rate to executing two consecutive  $SI$  iterations in the best case scenario. The  $P$ - $PI$ - $SI$  spectrum, however, does not exhibit a lower bound for  $c < 1$ . This comparison preliminarily suggests  $P$ - $PI$ - $SI$  to be a more effective method, having the ability to model a larger portion of a problem's transport phenomena with a single iteration than  $P$ - $IP$ - $SI$ .

### 4.3: Eigenvalues of $PBJ$ methods as functions of fundamental transport phenomena

In Fig. 4.3, we present the fundamental eigenvalue of  $PBJ$ - $ITMM$  as a function of scattering ratio and cell optical thickness. It is clear from this graph that  $PBJ$ - $ITMM$ 's iterative performance degrades as the scattering ratio increases, which is seemingly discrepant with our physical interpretation of a single  $PBJ$ - $ITMM$  iteration as the simulation of particles undergoing any number of collisions within a sub-domain, only ceasing simulation of particles once they cross a sub-domain boundary. With this interpretation, it is initially unclear why  $PBJ$ - $ITMM$ 's iterative performance degrades with increased scattering. To investigate this feature, we begin by recognizing that while there are two dependencies for the fundamental eigenvalue of  $PBJ$ - $ITMM$  (and  $IPBJ$ ) that may take multiple forms. In addition to the scattering ratio and optical thickness,  $\Sigma_a$  and  $\Sigma_s$  may be used to manifest this dependence, thus representing the fundamental eigenvalue as a function of the fundamental transport phenomena of absorption and scattering, respectively. The fundamental eigenvalues of  $PBJ$ - $ITMM$  and  $IPBJ$ , Eq. (3.33) or (3.47), respectively, divided by (3.40), are displayed below in Figs. 4.18 and 4.19 as functions of  $\Sigma_a$  and  $\Sigma_s$  for a problem with cells of unit size.

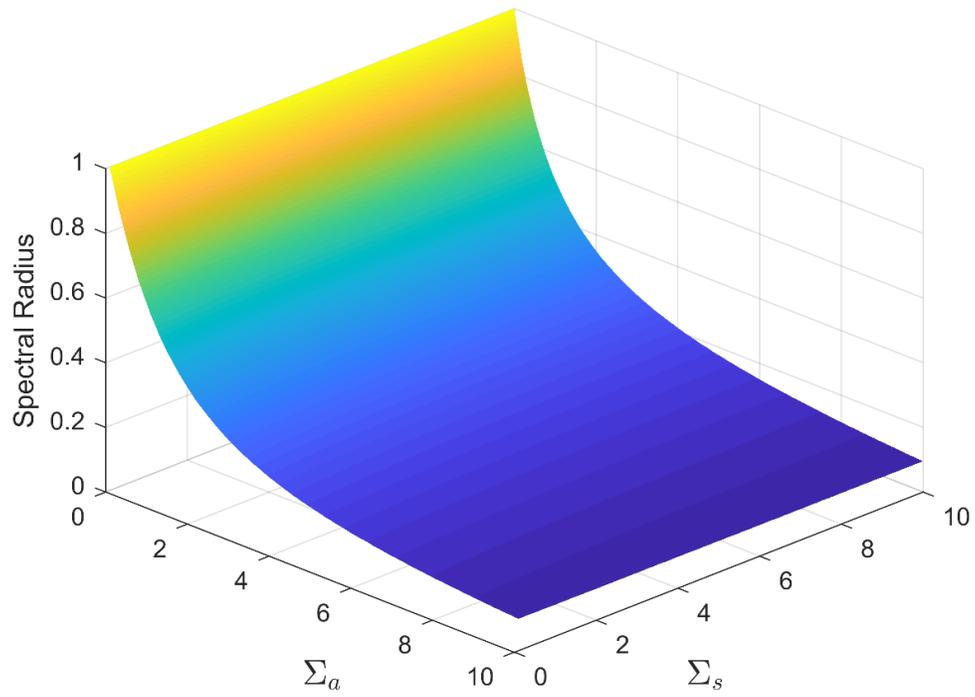


Figure 4.18: Spectral radius of *PBJ-ITMM* vs.  $\Sigma_a$  and  $\Sigma_s$  with  $S_6$  angular quadrature

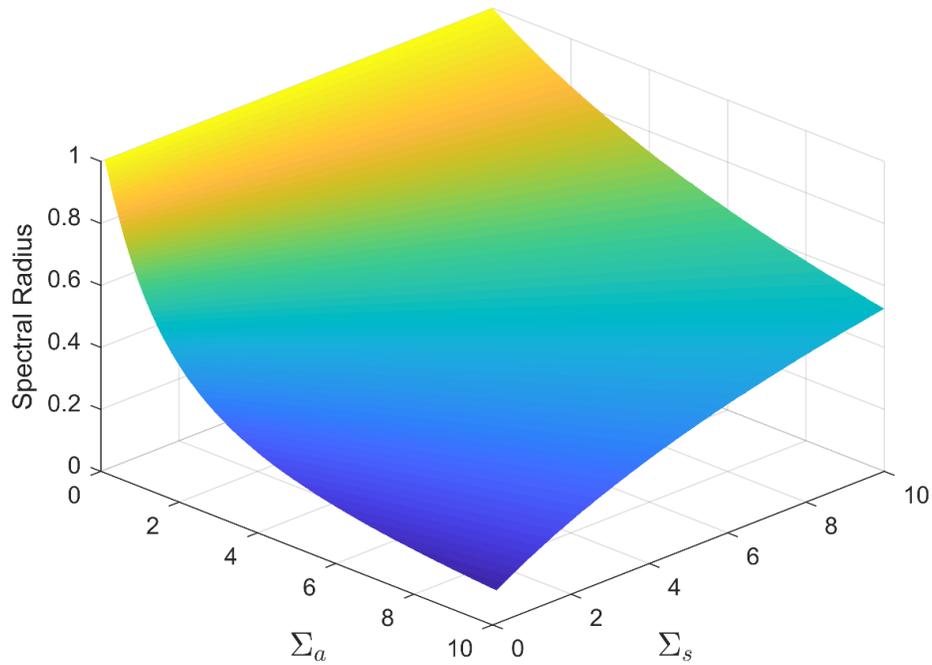


Figure 4.19: Spectral radius of *IPBJ* vs.  $\Sigma_a$  and  $\Sigma_s$  with  $S_6$  angular quadrature

Analyzing Fig. 4.18, we are able to reconcile our physical interpretation of *PBJ-ITMM* with the method's fundamental eigenvalue (spectral radius). It is apparent from this graph that the spectral radius of *PBJ-ITMM* is almost entirely determined by  $\Sigma_a$ , with the dependence on  $\Sigma_s$  being very weak. This indicates that, for a fixed value of  $\Sigma_t$ , the increase in *PBJ-ITMM*'s spectral radius with increasing  $c$  is not due to the increase in scattering, but rather, the decrease in absorption. Additionally, while the dependence on  $\Sigma_s$  is too weak to be easily discernable, the data on which Fig. 4.18 was produced demonstrates that, for a fixed value of  $\Sigma_a$ , *PBJ-ITMM*'s spectral radius actually decreases slightly with increasing  $\Sigma_s$ , a feature distinguishing *PBJ-ITMM* from all other iterative transport solution methods studied. This feature is explainable because, with *PBJ-ITMM* resolving local scattering within a sub-domain, its convergence rate is based entirely on the average number of sub-domain interfaces crossed by particles before loss via absorption (no leakage in the infinite medium associated with Fourier analysis). For a fixed value of  $\Sigma_a$ , the average distance traversed by particles before absorption is fixed. With increasing  $\Sigma_s$ , particles' flight paths become crooked and condensed, reducing the average number of sub-domain interfaces crossed for the same total distance traveled, and increasing *PBJ-ITMM*'s rate of convergence.

Contrarily, Fig. 4.19 shows *IPBJ* to be significantly dependent on both  $\Sigma_a$  and  $\Sigma_s$ . This is expected as the lagged scattering source in *IPBJ* results in the inability to resolve multiple collisions within a single iteration. The depiction of the spectra of *PBJ-ITMM* and *IPBJ* as functions of  $\Sigma_a$  and  $\Sigma_s$  rather than scattering ratio and cell size as previously presented allows for prediction of iterative performance trends based on fundamental transport phenomenon rather than mathematical quantities. In subsequent sections, this allows us to explain observations in non-model problems based on the problem's physical specifications. In general, this representation informs us that, for *PBJ-ITMM*, convergence rate depends almost solely on how far particles travel, on average, before being terminated, either by absorption or leakage. *IPBJ*'s convergence rate will depend on this, as well as the total number of collisions, on average, the particles undergo.

## 4.4: Comparison of *P-PI-SI* & *P-IP-SI* to their individual iterative constituents

*P-PI-SI* and *P-IP-SI* differ from previously discussed acceleration techniques in that they are combinations of two high-order iterative methods, each of which is capable of independently

converging the transport solution on its own. This presents an opportunity not usually available when studying acceleration methods; the ability to compare the combined method's performance to that of each individual method to determine the specific benefit of combining them. We begin this comparison with the spectral radius trends for  $P$ - $PI$ - $SI$  and  $P$ - $IP$ - $SI$  depicted in Figs. 4.16 and 4.17. We see that for either iterative method, the spectral radius approaches the scattering ratio as the cell size decreases. We also know the spectral radius of  $SI$  to equal the scattering ratio. This indicates that as the cell size decreases, the addition of  $PBJ$  iterations begin to have no effect on the rate of convergence.

To explain this phenomenon, we refer to our physical interpretations of the  $P$ - $PI$ - $SI$  iterative steps, that the  $PBJ$ - $ITMM$  iteration simulates particles scattering within a sub-domain until they exit, and an  $SI$  iteration simulates particles streaming across the medium until they collide. With this in mind, for particles which stream through a sub-domain, the  $PBJ$ - $ITMM$  iteration produces no additional information, as the  $SI$  iteration would have accounted for this uncollided streaming phenomena. Therefore, as the cell size decreases and the fraction of particles which travel uncollided through a sub-domain tends towards unity, the  $PBJ$ - $ITMM$  iteration's contribution to the combined iterative process diminishes and the convergence becomes driven solely by  $SI$ .

For  $P$ - $IP$ - $SI$ , while the interpretation of the  $IPBJ$  step differs from that of  $PBJ$ - $ITMM$ , the result is the same. The  $IPBJ$  step accounts for particles traveling within a sub-domain until they either leave the sub-domain or undergo a collision. For streaming, the dominant transport phenomenon in optically thin cells, there is nothing modeled by  $IPBJ$  that would not have been modeled with  $SI$ , leaving the convergence rate of  $P$ - $IP$ - $SI$  equivalent to that of  $SI$  alone.

Analogously, as the cell size increases, the spectral radius of  $P$ - $PI$ - $SI$  approaches that of  $PBJ$ - $ITMM$ . Based on the physical interpretation of the iterative sequence,  $SI$  contributes little to the simulation of particles which enter a sub-domain and collide before exiting, as this event would be more fully simulated by the  $PBJ$ - $ITMM$  iteration. Therefore, as the cell size increases and the fraction of particles entering a sub-domain which then interact within that sub-domain approaches unity, the  $SI$  iteration contributes minimally to the combined iterative sequence, causing the  $P$ - $PI$ - $SI$  spectral radius to tend toward that of  $PBJ$ - $ITMM$ .

In optically thick cells, the scenario for  $P$ - $IP$ - $SI$  is different. As the optical thickness increases, the  $P$ - $IP$ - $SI$  spectrum tends towards  $c^2$ , which is equivalent to the square of either  $SI$ 's

or *IPBJ*'s spectral radius in optically thick cells. This indicates that *IPBJ* and *SI* become comparable in optically thick cells from the standpoint of iterative convergence rate. While the contribution of *SI* to modeling collision-dominated optically thick regimes with *P-IP-SI* does not disappear as was the case with *P-PI-SI*, the overall benefit of using *P-IP-SI* as opposed to *IPBJ* is eliminated, as two *IPBJ* iterations would provide approximately the same reduction in error without the penalty to parallelism from *SI*'s sequential nature.

We demonstrate this concept by plotting the number of iterations required by HAT-2C to converge a  $100 \times 100$  homogeneous-material problem using *SI*, *PBJ-ITMM*, *IPBJ*, *P-PI-SI*, and *P-IP-SI* versus cell size for  $c = 0.1, 0.5,$  and  $0.9$  in Figs. 4.20 - 4.22, respectively. For these numerical experiments, an  $S_6$  angular quadrature was used with a relative stopping criterion of  $10^{-6}$ .

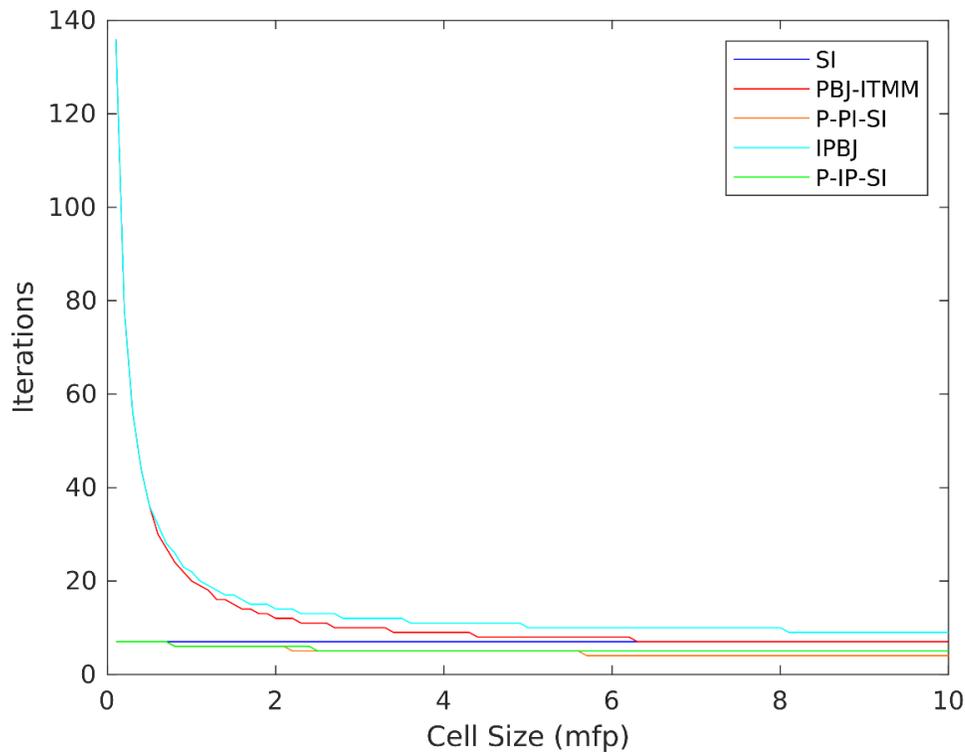


Figure 4.20: Iterations required to converge  $100 \times 100$  homogeneous problem:  $c = 0.1$

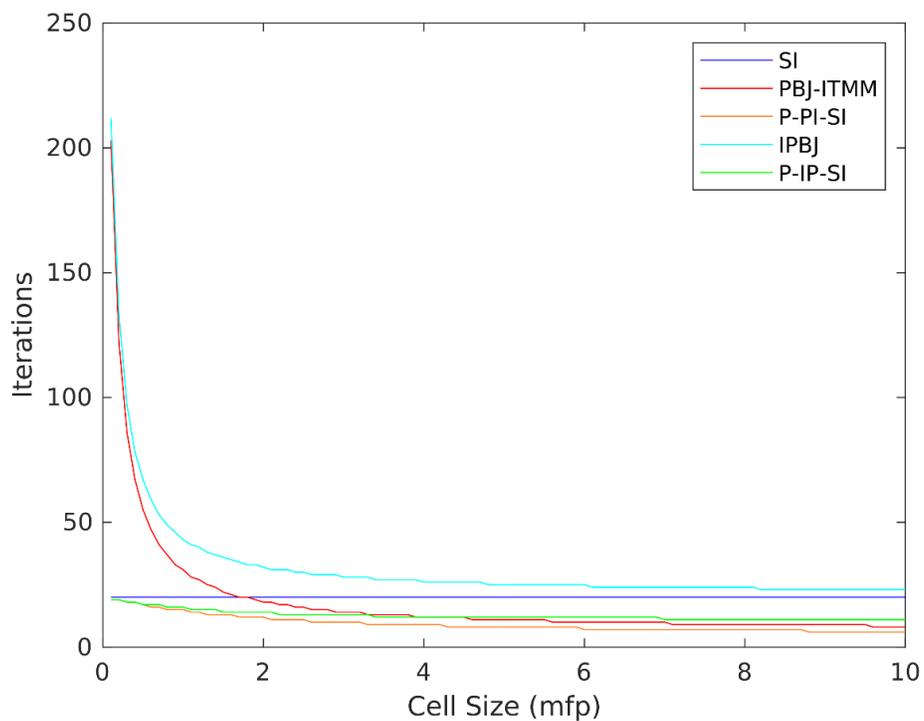


Figure 4.21: Iterations required to converge  $100 \times 100$  homogeneous problem:  $c = 0.5$

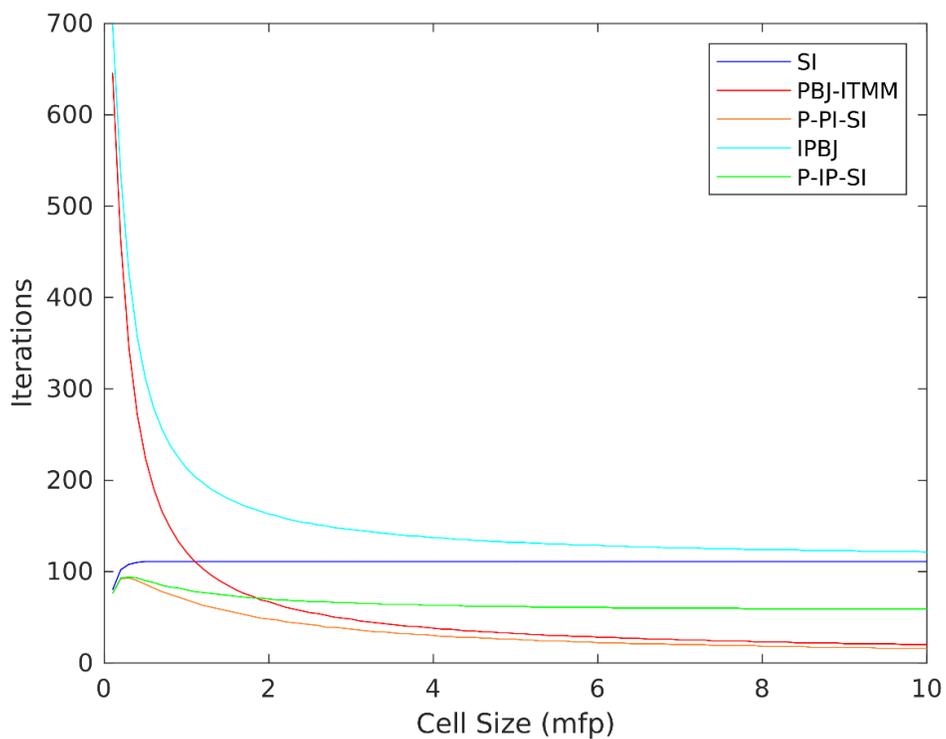


Figure 4.22: Iterations required to converge  $100 \times 100$  homogeneous problem:  $c = 0.9$

From the trends observed in Figs. 4.20 - 4.22 the predictions made from the Fourier analysis are confirmed. In the optically thin regime, the number of required iterations for  $P$ - $PI$ - $SI$  tends towards that of  $SI$ , and towards that of  $PBJ$ - $ITMM$  in the thick regime. In the optically thin regime, the number of required iterations for  $P$ - $IP$ - $SI$  also tends towards that of  $SI$ , and towards  $\frac{1}{2}$  of  $SI$  or  $IPBJ$ 's required iterations in the optically thick regime. Note that these trends are easiest to observe in the  $c = 0.9$  case, as the difference in iteration count trends is far more pronounced.

These graphs demonstrate that  $P$ - $PI$ - $SI$  converges problems due only to the individual method that is most suited for the problem's cell size, with the other method doing very little to assist in convergence for homogeneous problems with most values of cell optical thickness. Similarly,  $P$ - $IP$ - $SI$  converges almost exclusively due to  $SI$  in the optically thin regime and both  $IPBJ$  and  $SI$  equally in the optically thick regime. This is not due to the  $SI$  preconditioning approach being more effective for  $IPBJ$  though. Contrarily, this is because  $IPBJ$  sees its convergence rate approach that of  $SI$  in the optically thick regime, as the effect on convergence rate of lagging the incoming angular fluxes diminishes. Therefore,  $P$ - $IP$ - $SI$  converges problems in the optically thick regime using both  $IPBJ$  and  $SI$  because  $IPBJ$  is unable to achieve the unboundedly fast convergence rates of  $PBJ$ - $ITMM$  in this regime. These results further confirm our claim about the underlying mechanism of the proposed hybrid approach, but they also indicate that a more ideal hybrid approach exists, as using either  $SI$  preconditioning method outside of a narrow regime that lies between the thin and thick regimes would impose increased computational cost per iteration and a reduction in the degree of parallelism for only a small increase in convergence rate.

## **4.5: $P$ - $PI$ - $SI$ & $P$ - $IP$ - $SI$ in Heterogeneous Problems and the Asynchronous Hybrid Approach**

Given the previous comparison of  $P$ - $PI$ - $SI$  and  $P$ - $IP$ - $SI$  to the iterative methods which constitute their individual steps for homogeneous media, the methods appear to be advantageous only for very specific ranges of problem parameters. We proposed  $SI$  preconditioning in order to make  $PBJ$ - $ITMM$  robust in optically thin cells, which it accomplished, but as demonstrated, in optically thin cells, unaccelerated  $SI$  is actually a better option. So far though, we have only shown results for homogeneous problems, as necessitated to facilitate comparison to the theoretical analysis. Almost any real-world problem of interest will involve heterogeneous material

composition, where we expect  $P-PI-SI$  to be a more effective iterative scheme. If a problem contains both optically thin and thick cells, then our physical interpretation of the iterations suggests that the  $PBJ-ITMM$  step will allow the numerous local collisions in the optically thick cells to be resolved saving many  $SI$  iterations, and particles can be modeled streaming across the optically thin cells in a single  $SI$  iteration, saving the many  $PBJ-ITMM$  iterations required for this to be modeled. With our interpretation of  $IPBJ$  though, we expect  $P-IP-SI$  to be less effective, as  $IPBJ$  in thick cells only sees the negative impact of lagging the sub-domain boundary incoming angular fluxes diminished. Unlike  $PBJ-ITMM$ ,  $IPBJ$  does not improve iterative robustness over  $SI$  for modeling the repeated localized collisions prevalent in optically thick cells.

To test these conjectures, we developed a periodic vertical interface problem [22] with both optically thick cells and optically thin cells. The geometry of this problem is shown in Fig. 4.23. A  $100 \times 100$  cell mesh is imposed on this problem configuration, making each stripe 10 cells wide. Each cell within the thick regions is  $10 \text{ mfp}$  thick and each cell within the thin regions is  $0.1 \text{ mfp}$  thick. All boundary conditions are vacuum. We observe the required number of iterations consumed by HAT-2C to converge this problem to a relative stopping criterion of  $10^{-6}$  using  $SI$ ,  $PBJ-ITMM$ ,  $IPBJ$ ,  $P-PI-SI$ , and  $P-IP-SI$  with  $AHOT-N0$  for various scattering ratios, employing  $S_6$  angular quadrature.

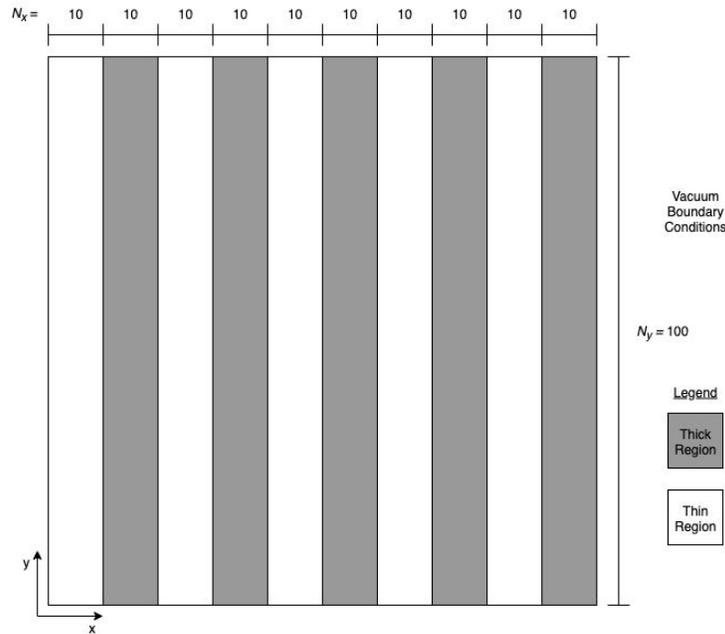


Figure 4.23: Layout of heterogeneous stripe problem with uniform unit source. Thick regions contain  $10 \text{ mfp}$  cells, thin regions contain  $0.1 \text{ mfp}$  cells.

In addition to these five iterative methods, we also introduce two more variants in these numerical experiments, namely the *Asynchronously Hybrid Parallel Block Jacobi – Integral Transport Matrix Method / Source Iteration (AH-PI-SI)* and the *Asynchronously Hybrid Inexact Parallel Block Jacobi / Source Iteration (AH-IP-SI)* methods. *AH-PI-SI* and *AH-IP-SI* differ from *P-PI-SI* and *P-IP-SI*, respectively, in that they run only *SI* in thin cells and only the associated *PBJ* method in thick cells. The asynchronously hybrid method class was conceived based on the previous observation that one of the two individual iterative methods constituting either of the preconditioner methods were found to be more computationally advantageous, with the dominant method dependent on the optical thickness of cells in a given problem. We predicted that this effect would extend to a heterogeneous problem, meaning that with *P-PI-SI*, little improvement would be realized in the rate of convergence by running *PBJ-ITMM* in optically thin cells or by running *SI* in optically thick cells. For *P-IP-SI*, *SI* would still offer improvement of convergence rate in optically thick cells, but no more than achievable by running a second *IPBJ* iteration instead, thus rendering it an unnecessary penalty to the degree of parallelism. *AH-PI-SI* and *AH-IP-SI* were introduced as a means to reduce the cost of an iteration and increase the degree of parallelism of the solution with limited impact on the required number of iterations. The exchange of information between the iteration steps in the asynchronous hybrid methods is different than in the preconditioner methods as well. Since a *PBJ* method and *SI* are not performed in the same cells, only the angular fluxes on the interface between thick and thin regions are exchanged in between the two iterative methods.

While it is trivial to see how the asynchronously hybrid approach reduces the cost of an iteration over the preconditioning approach, as it only runs one iterative step in each cell as opposed to two, it may be less obvious how it increases the degree of parallelism. Recall that we originally recognized that one of the main sacrifices made when developing *P-PI-SI* is the fact that the *SI* step would be required to run in spatially serial operation, as we are investigating *PBJ-ITMM* as an ultimate means by which to perform massively parallel transport calculations in unstructured meshes as an alternative to the complex sweep algorithms required for solution using *SI*. If we consider the problem represented by Fig. 4.23, only 10% of the problem must be run in this serial manner, as all 5 of the *SI* zones may be executed in parallel with all other zones, whether *SI* or *PBJ*.

Formally, the *asynchronous hybrid* methods employ a new type of spatial domain decomposition, which we will refer to as a *Hybrid Spatial Domain Decomposition (HSDD)*. An *HSDD* splits the spatial domain into multiple regions (or zones), each of which employs a local spatial domain decomposition based on the optical thickness of the cells comprising it. Zones with cells thinner than a specified threshold employ standard *SI* iterations, while cells thicker than this threshold employ *PBJ* iterations.

An *HSDD* has lagged angular fluxes on zone interfaces. This is the source of the additional asynchronicity of *AH-PI-SI* and *AH-IP-SI*, with zones decoupled from each other over the course of an iteration. This is reminiscent of the sub-domain decomposition used with *PBJ*, but on a larger scale. This asynchronous aspect of the *HSDD* is what allows iterative solutions within individual zones using different methods at the same time, thus restoring much of the degree of parallelism that was originally sacrificed with *P-PI-SI* and *P-IP-SI*.

The number of iterations required to converge the test problem depicted in Fig. 4.23 for *SI*, *PBJ-ITMM*, *P-PI-SI*, *AH-PI-SI*, *IPBJ*, *P-IP-SI*, and *AH-IP-SI* are shown in Table 4.5.

Table 4.5: Number of iterations required to converge the heterogeneous striped problem depicted in Fig. 4.23 with increasing  $c$ , using the various iterative strategies and *AHOT-N0*. Relative iteration stopping criterion:  $10^{-6}$ ,  $S_6$  angular quadrature. *PBJ* methods use one cell per sub-domain.

$c$	<i>SI</i>	<i>PBJ-ITMM</i>	<i>P-PI-SI</i>	<i>AH-PI-SI</i>	<i>IPBJ</i>	<i>P-IP-SI</i>	<i>AH-IP-SI</i>
0.1	8	68	6	8	70	6	10
0.2	10	75	8	10	77	8	13
0.3	13	80	9	11	83	9	16
0.4	17	86	11	13	90	11	20
0.5	21	94	13	15	99	13	25
0.6	28	103	15	17	111	17	32
0.7	39	115	17	20	127	22	44
0.8	59	133	21	25	152	33	66
0.9	116	167	29	35	213	63	129

From the results reported in this table, we first confirm our predictions regarding the performance of *P-PI-SI*, as we see a significant improvement over each of the individual methods

used. As we have discussed, we attribute this to the fact that the two iterative methods do not rely on one another to converge the solution in parts of a problem where the other iterative method is ineffective. We can see from the required number of *PBJ-ITMM* iterations that optically thin cells within a problem cause significant iterative slowdown even if the entire domain is not optically thin. *SI* preconditioning is found to dramatically reduce this slowdown by allowing *SI* to simulate the streaming of particles through the optically thin regions and *PBJ-ITMM* to simulate the local scattering of particles within the optically thick regions. Since most realistic problems will contain cells of a variety of optical thicknesses, we conclude that *SI* preconditioning is an effective method for alleviating the iterative slowdown *PBJ-ITMM* experiences due to optically thin cells, as it shows a significant reduction in the number of required iterations over each of its individual constituent iterative methods.

With the concept of combining the *PBJ-ITMM* method with the *SI* method proven to be effective at obtaining solutions without dramatic iterative slowdown in optically thin cells, we now discuss the results for *AH-PI-SI*, which attempts to combine these two methods in a more effective manner than simply running both methods sequentially in all cells. We see from Table 4.5 that *AH-PI-SI* requires, at most, 6 more iterations than *P-PI-SI* for all tested problems. For the given test problem configuration, *AH-PI-SI* requires about half of the computations per iteration that *P-PI-SI* requires (the exact reduction in number of computations required per iteration will be problem dependent). With this consideration alone, *AH-PI-SI* definitively obtains this problem's solution in shorter execution time than *P-PI-SI*. Furthermore, *AH-PI-SI* restores much of the degree of parallelism that was originally sacrificed when using *P-PI-SI*. For these reasons, we predict *AH-PI-SI* will be a more effective method for combining *PBJ-ITMM* and *SI* than *P-PI-SI*, obtaining practically the same improved convergence rate in optically thin cells with a reduced iteration cost as measured by execution time and an increased degree of parallelism.

For *P-IP-SI*, we similarly see a reduction in the number of required iterations over *SI* and *IPBJ* individually. At high scattering ratios though, the reduction in the number of iterations compared to *SI* is less dramatic than was the case with *P-PI-SI*. For the  $c = 0.9$  case, the number of *P-IP-SI* iterations required is over twice the number required for *P-PI-SI*, and in fact, is roughly  $\frac{1}{2}$  the number required for *SI*. Then, moving to the asynchronous hybrid approach, *AH-IP-SI* is seen to have a much larger increase in iterations over *P-IP-SI* than was the case with the *PBJ-ITMM* counterparts. All of this is indicative of the understanding that *IPBJ* in optically thick cells

becomes equivalent to *SI* from the standpoint of convergence rate. Consequently, *IPBJ / SI* hybrid methods incur a strict limitation on their convergence rate precipitated by a fundamental transport phenomenon, specifically repeated localized scattering, without an effective mechanism for its modeling.

## 4.6: Parametric Study of Hybrid Methods

With *AH-PI-SI* demonstrating potential as a method which balances applicability in massively parallel HPC environments with improved convergence rate, we perform a parametric study using variations of the heterogeneous stripe problem depicted in Fig. 4.23. This study is designed to assess the iterative performance of hybrid methods in comparison to other hybrid combinations, their individual iterative components, and traditional acceleration methods as a function of four independent variables: (1) the number of cells in a stripe,  $N_s$ ; (2) optical thickness of the thin and (3) thick cells; and (4) the scattering ratio. The metrics used to assess iterative effectiveness are: (1) the number of iterations required for convergence; and (2) the total execution time. By observing the effect of these independent variables on the iterative effectiveness of the various iterative methods, we generalize these effects to ranges of problem parameters and geometry configurations. Note that *pNDA*, not *NDA* results are presented for this parametric study and subsequent studies that present results for traditional acceleration methods. This is because *NDA* was observed to be iteratively unstable, with its solution divergent for many of the problems tested in these experiments. In contrast, the implementation of *pNDA* in HAT-2C, is observed to be stable for the parametric study problem configuration.

While Fig. 4.23 depicts the general configuration of the problem used for our parametric study, changes to the geometry were made to accommodate the varying nature of selected parameters within the range investigated in this study. Angular quadrature and relative stopping criterion remain unchanged throughout, however, the optical thickness of cells in either region, the number of cells per stripe, and the total number of stripes are all subject to change. Additionally, in previous sections of this chapter, *PBJ* methods utilized single cell sub-domains, as this was the case considered in the theoretical analysis. Previous work, however, has established that the optimal sub-domain size for *PBJ-ITMM* in 3-D geometry is  $4 \times 4 \times 4$  cells. [55] While the optimal size is ultimately problem and HPC dependent, this is clearly a feasible sub-domain size for

practical implementation. Therefore, in the parametric study,  $4 \times 4$  cells per sub-domain are used to mimic the convergence rates of the aforementioned 3-D implementation. *IPBJ* also uses this size of sub-domains to compare *PBJ* methods that possess the same theoretical scalability. To allow stripes to contain a uniform number of cells when *PBJ* sub-domains are of size  $4 \times 4$ , the mesh size is increased to  $128 \times 128$ .

Finally, in addition to the hybrid methods previously developed, we introduce another for this parametric study, *AH-PI-IP*, a hybrid method that implements *PBJ-ITMM* and *IPBJ* in regions containing optically thick and thin cells respectively. *AH-PI-IP* was developed based on the observation that the iterative performance of a *PBJ*-type method is ultimately dependent on the optical thickness of sub-domains rather than the optical thickness of individual cells (these two quantities were the same in previous studies that employed single-cell sub-domains). With *IPBJ*'s iterative solution time growing linearly with sub-domain size, it was theorized that *IPBJ* could be executed in optically thin regions in lieu of *SI*, with the optical thickness of sub-domains increased by increasing the number of cells per sub-domain to quantities that would likely render *PBJ-ITMM* iterations too costly, either from the standpoint of iterative solution time or memory requirement. Specifically, in the *AH-PI-IP* formulation, we constrain these *IPBJ* sub-domains to be of a size such that their per-angle iterative solution time is not longer than the iterative solution time of a single *PBJ-ITMM* sub-domain. Since the *IPBJ* solution can be angularly parallelized in our target application, this results in a hybrid method that has the same potential degree of parallelism in unstructured grids as *PBJ-ITMM*, fully eliminating the primary shortcoming originally associated with our hybrid approach. For this parametric study, to determine the size of *IPBJ* sub-domains for *AH-PI-IP*, we determined the per-sub-domain iterative computation time required by *PBJ-ITMM* for an  $8 \times 8$  sub-domain (comparable iterative solution time to  $4 \times 4 \times 4$  sub-domain in 3-D) and found this to be about  $21^3$  times the *SI* grind time (per-angle solution time for a single cell). The maximum number of cells in an *IPBJ* sub-domain for *AH-PI-IP* was therefore designated to be  $21^2$ , obtaining equivalent convergence rates to the 3-D counterpart. *IPBJ* sub-domains when using *AH-PI-IP* are sized accordingly in our studies.

For readability, only graphs necessary to adequately illustrate general trends are shown in the main text. For a complete set of the results of the parametric study, refer to Appendix A. To infer the primary findings of the parametric study, we first show the results for the case of 16 cells in the  $x$  direction per stripe ( $N_s$ ),  $\Sigma_{thin}\Delta u_{thin} = 0.01$ , and  $\Sigma_{thick}\Delta u_{thick} = 5.0$ . This is referred to

as the base case, with subsequent graphs then modifying one (or more) of these parameters, leaving all others fixed at their base value to observe the effect of that (those) parameter(s) on the iterative methods' convergence cost measured in number of iterations and runtime for all iterative methods considered. The full set of results from this parametric study in Appendix A provide additional evidence for the conclusions drawn from this study. The results of this parametric study were obtained using the HAT-2C code, with *ITMM* construction performed using *GFIC* with a full sweep. The reported execution times were measured on an Intel Xeon® E5-2630 v3 processor operating at 2.4 GHz with a Linux operating system using the gfortran compiler.

To effectively present this large amount of data, we use the following organizational tools. When reading any individual graph, the curves and legend are color coordinated such that all acceleration methods are varying shades of green, all methods containing *PBJ-ITMM* or *IPBJ* are shades of blue or red respectively (except for *P-PI-IP*). The *PBJ* method by itself is the darkest shade, with the *SI* preconditioned method a lighter shade, and the *asynchronous hybrid* combination with *SI* a lighter shade still. When comparing separate cases to the base case, in the figure caption, any parameter that is decreased by two data points below the base case are colored **dark red**; one data point below, **light red**; one data point above, **light blue**; and two data points above, **dark blue**. This results in the color code,  $N_s = [4, 8, 16, 32, 64]$ ,  $\Sigma_{thin}\Delta u_{thin} = [0.001, 0.01, 0.1, 0.5]$ , and  $\Sigma_{thick}\Delta u_{thick} = [1, 2.5, 5, 10]$ .

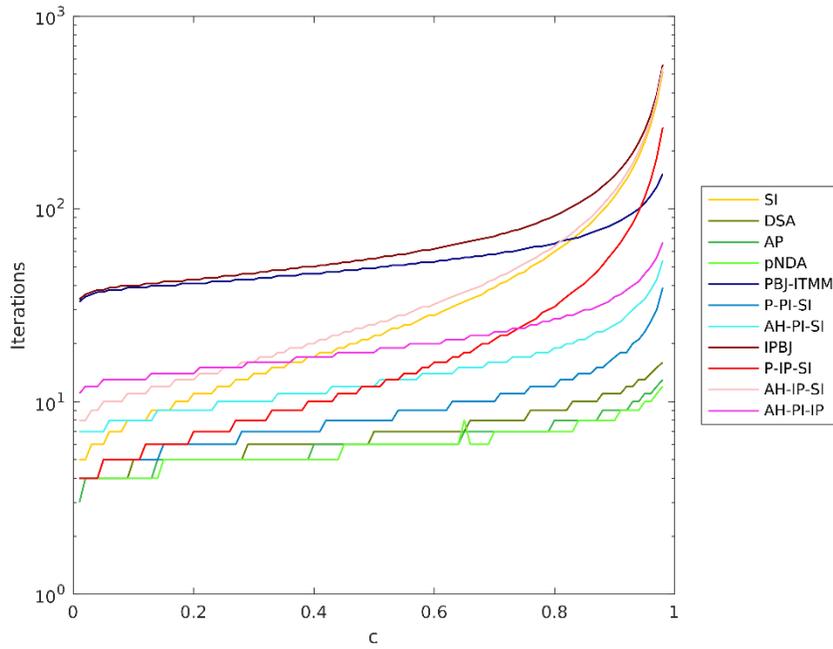


Figure 4.24: Iterations required to converge parametric study problem versus scattering ratio:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$  (Base Case)

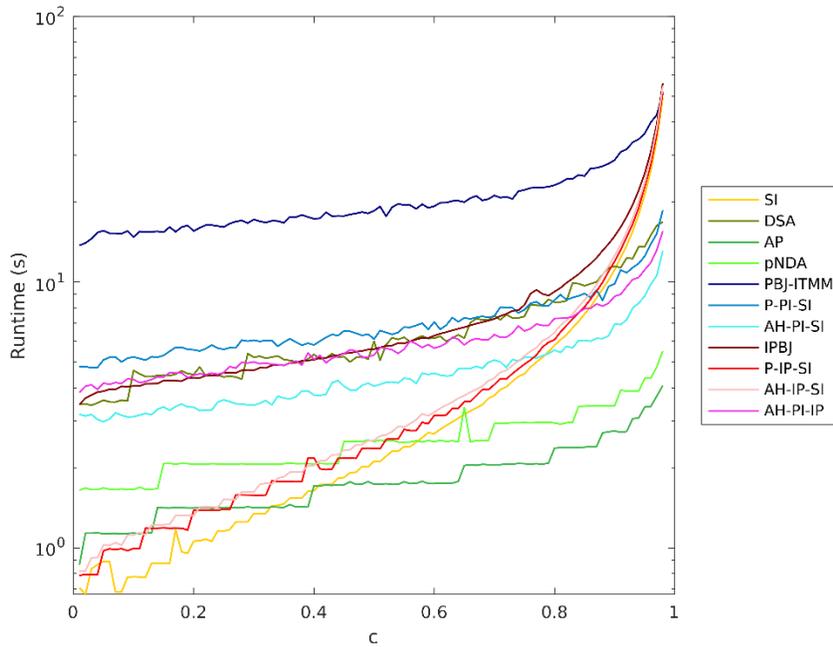


Figure 4.25: Runtime observed for parametric study problem versus scattering ratio:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$  (Base Case)

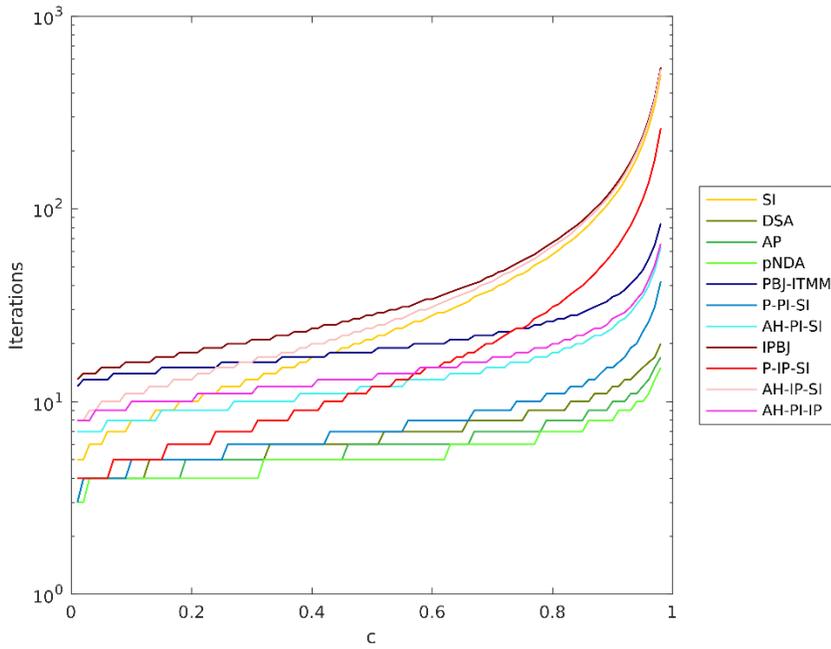


Figure 4.26: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 4, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$

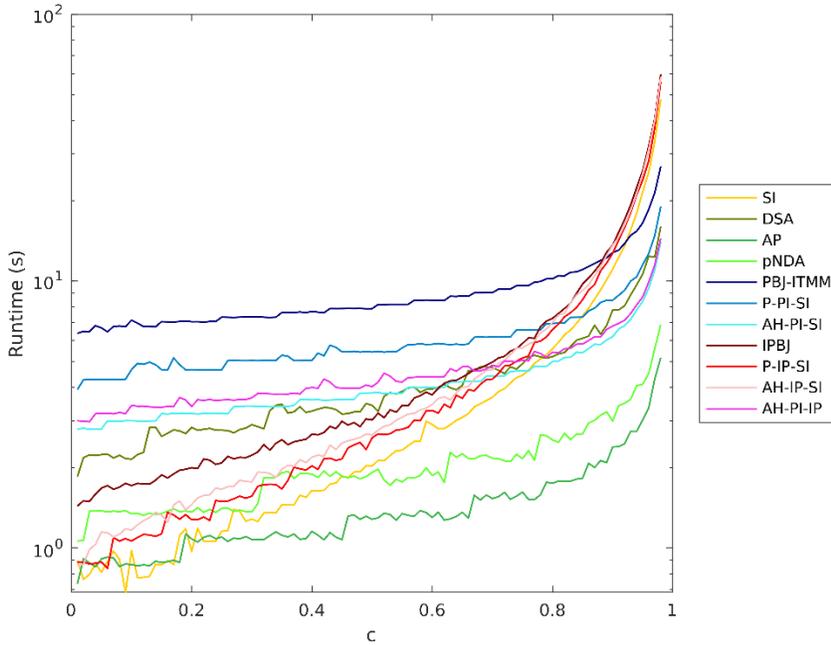


Figure 4.27: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 4, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$

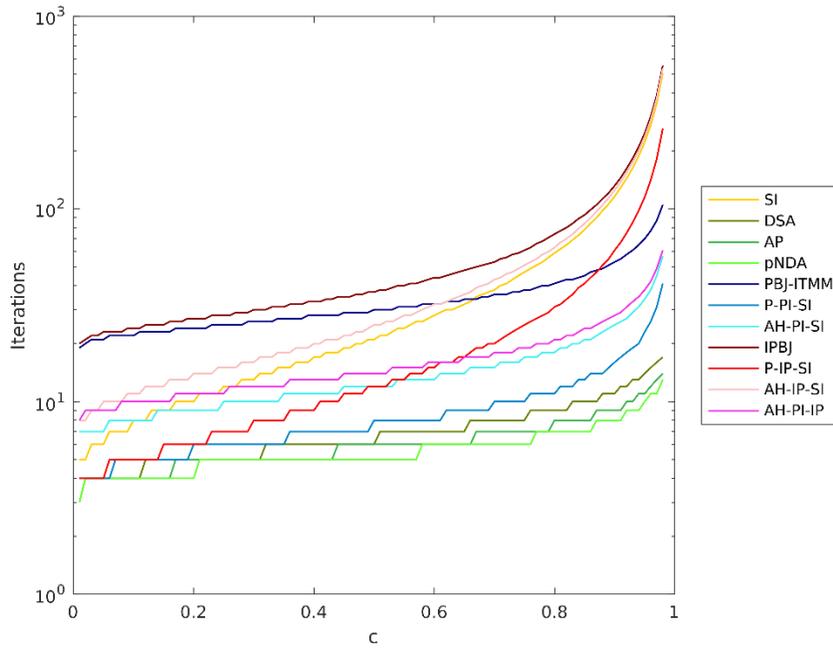


Figure 4.28: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 8, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$

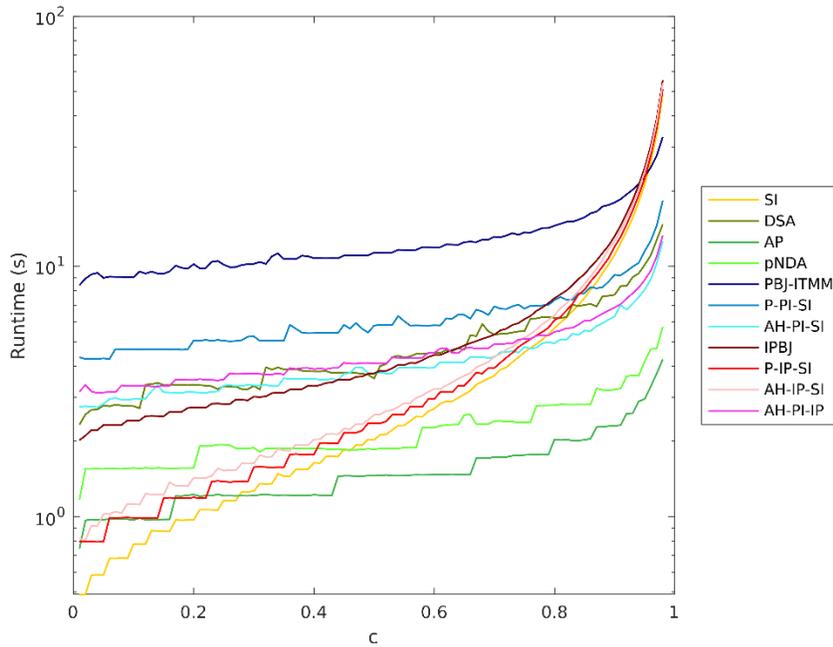


Figure 4.29: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 8, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$

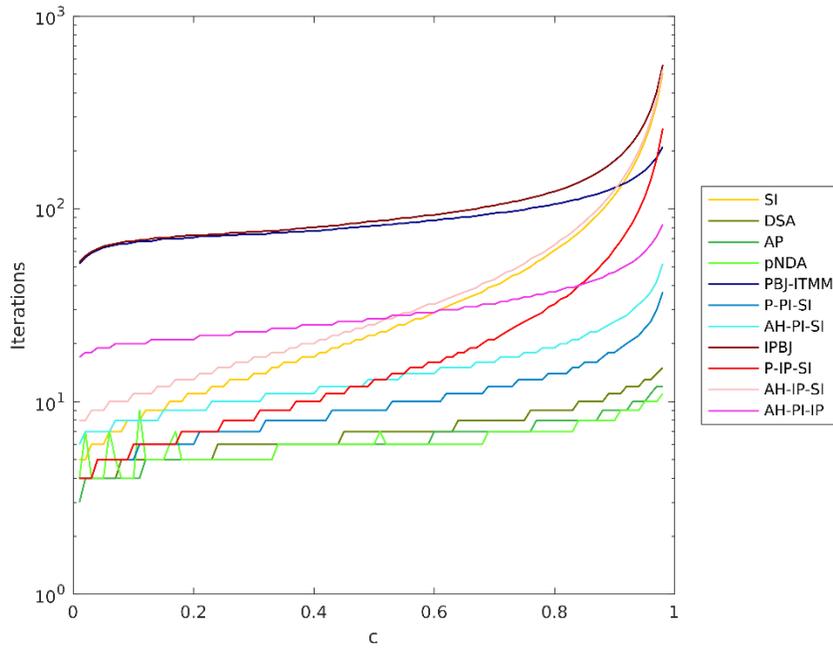


Figure 4.30: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 32, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$

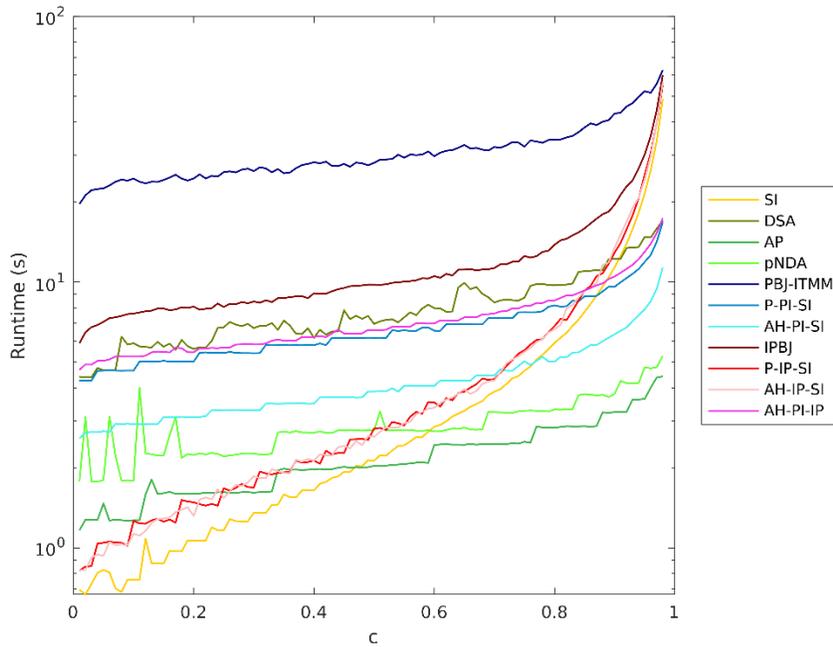


Figure 4.31: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 32, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$

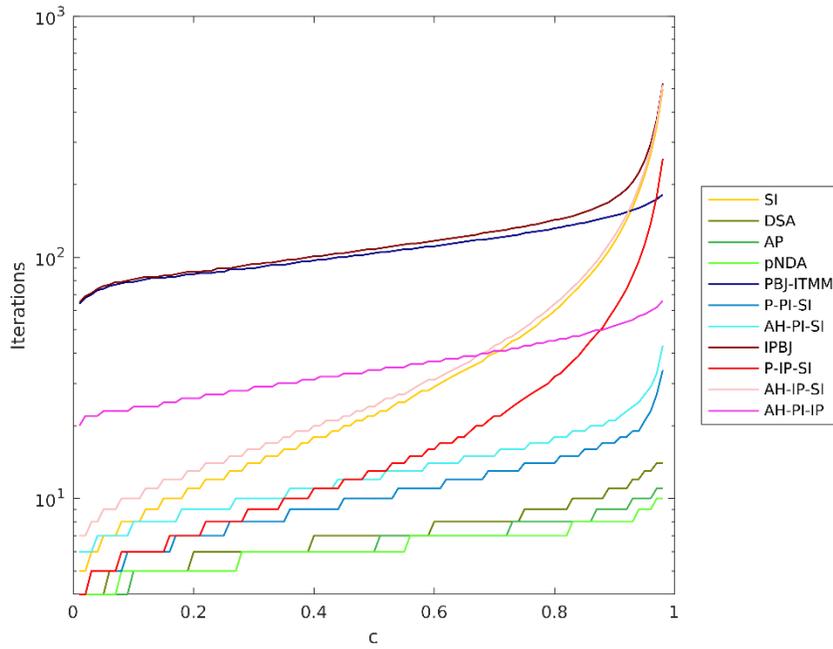


Figure 4.32: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 64, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$

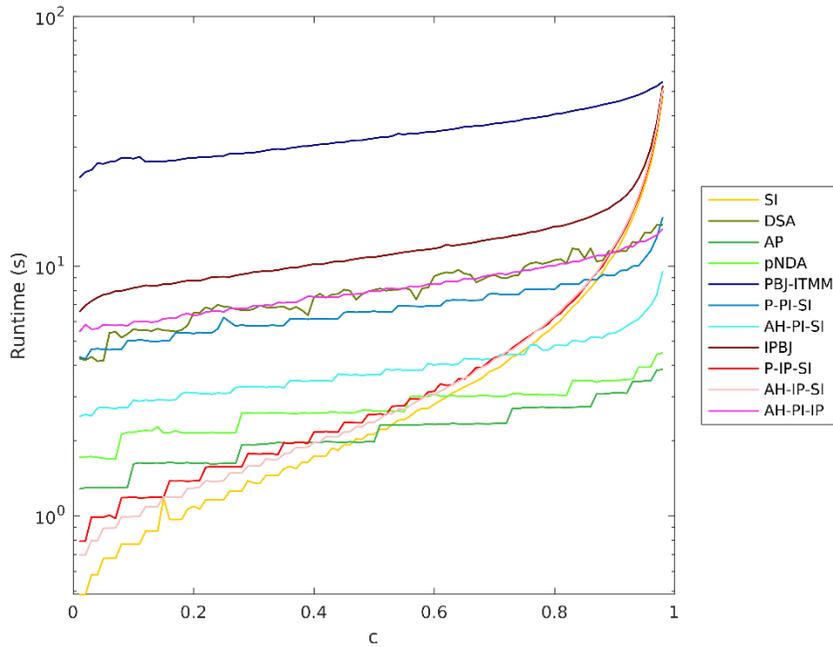


Figure 4.33: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 64, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$

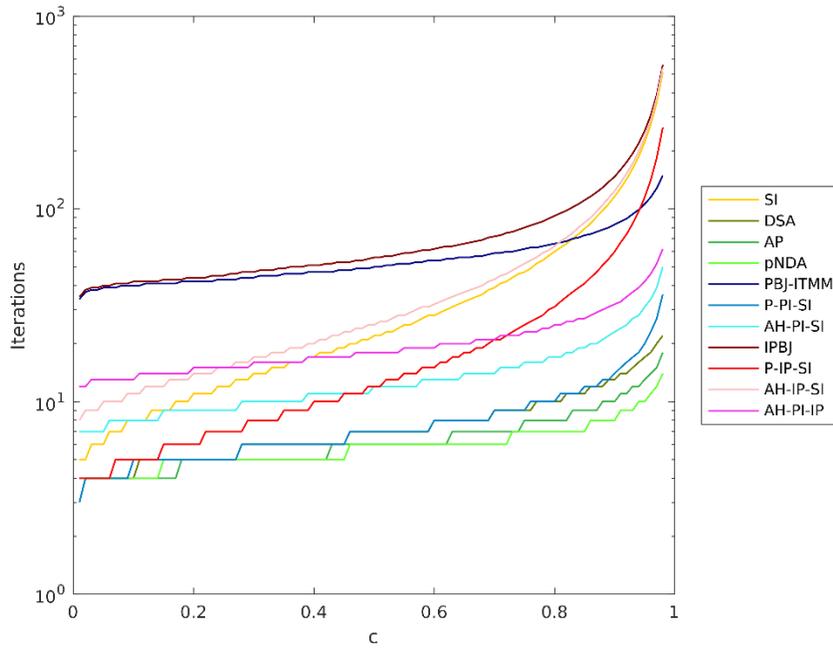


Figure 4.34: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.001, \Sigma_{thick} \Delta u_{thick} = 5$

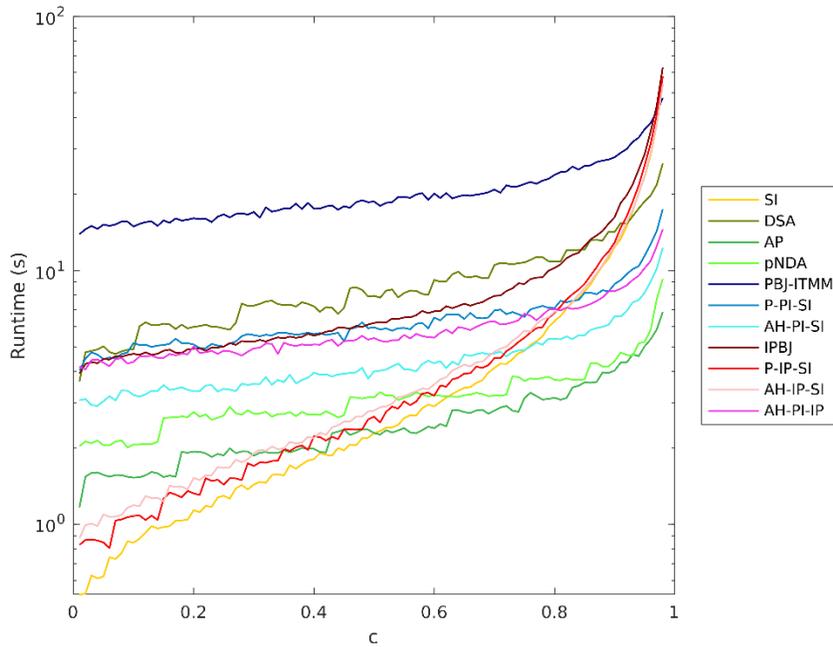


Figure 4.35: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.001, \Sigma_{thick} \Delta u_{thick} = 5$

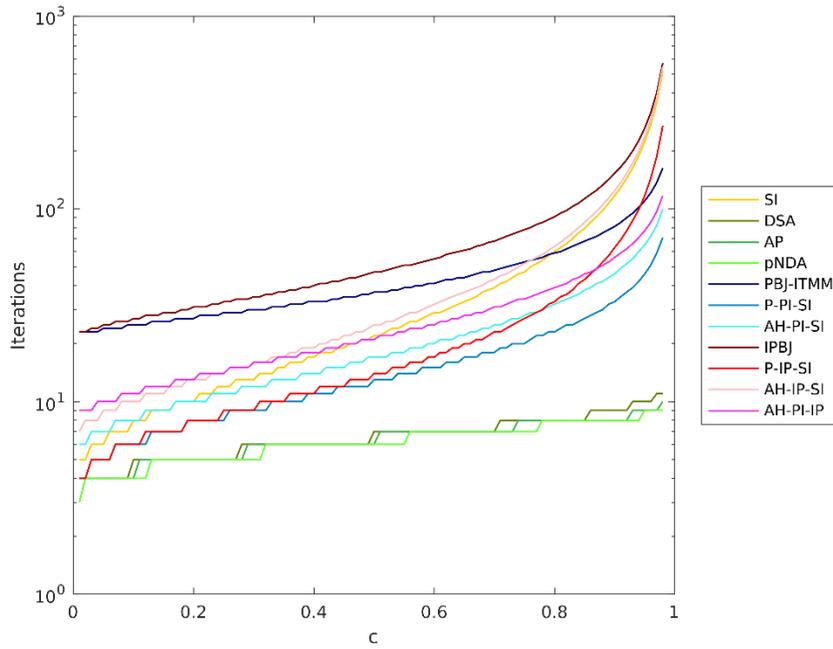


Figure 4.36: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.1, \Sigma_{thick} \Delta u_{thick} = 5$

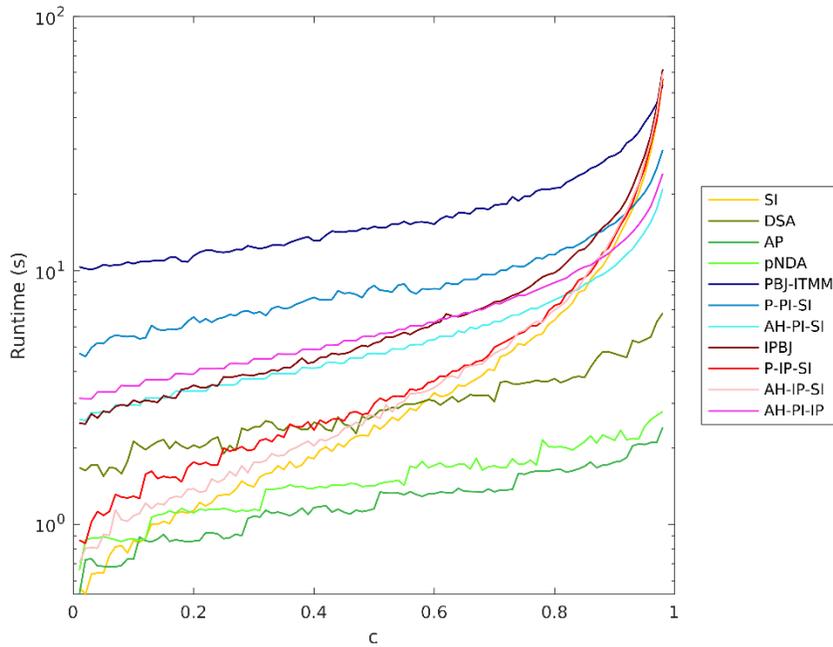


Figure 4.37: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.1, \Sigma_{thick} \Delta u_{thick} = 5$

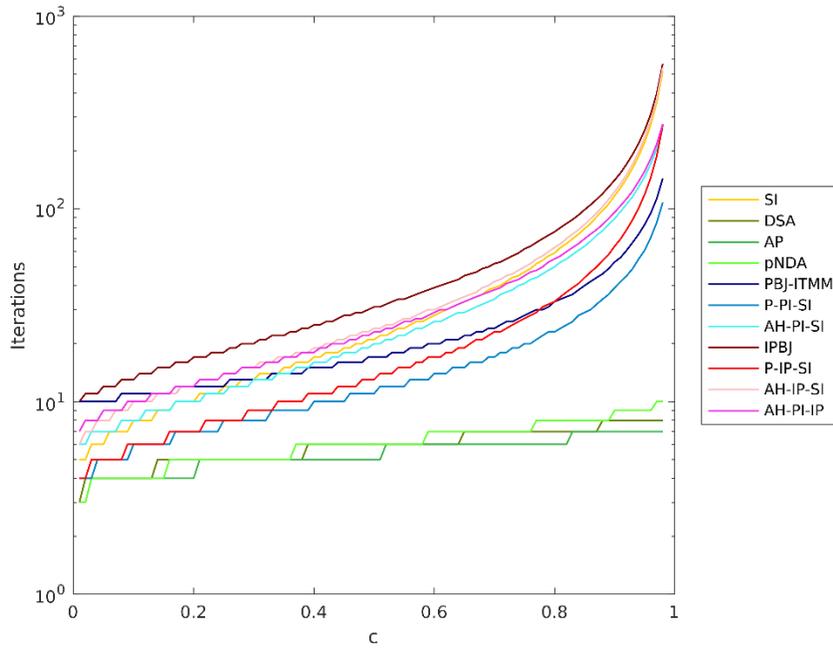


Figure 4.38: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.5, \Sigma_{thick} \Delta u_{thick} = 5$

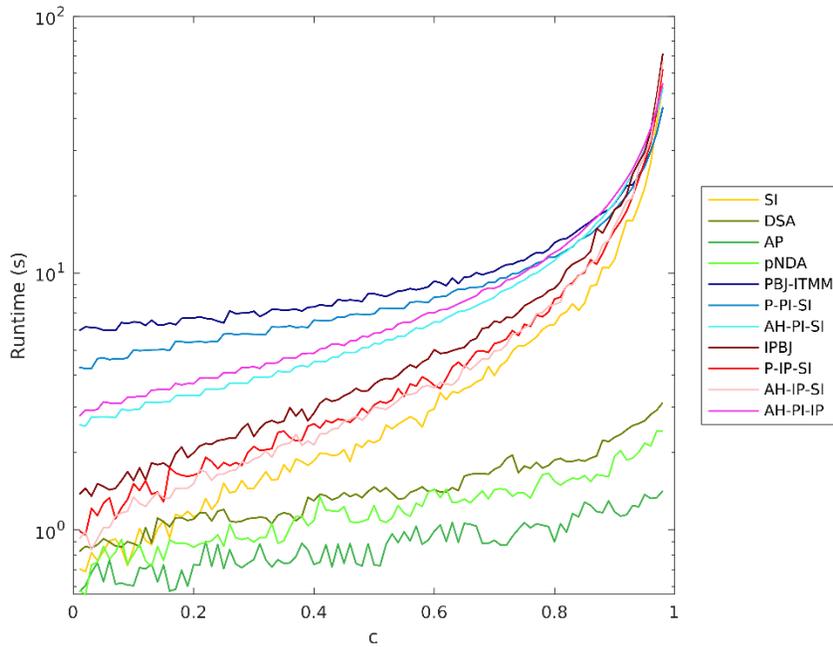


Figure 4.39: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.5, \Sigma_{thick} \Delta u_{thick} = 5$

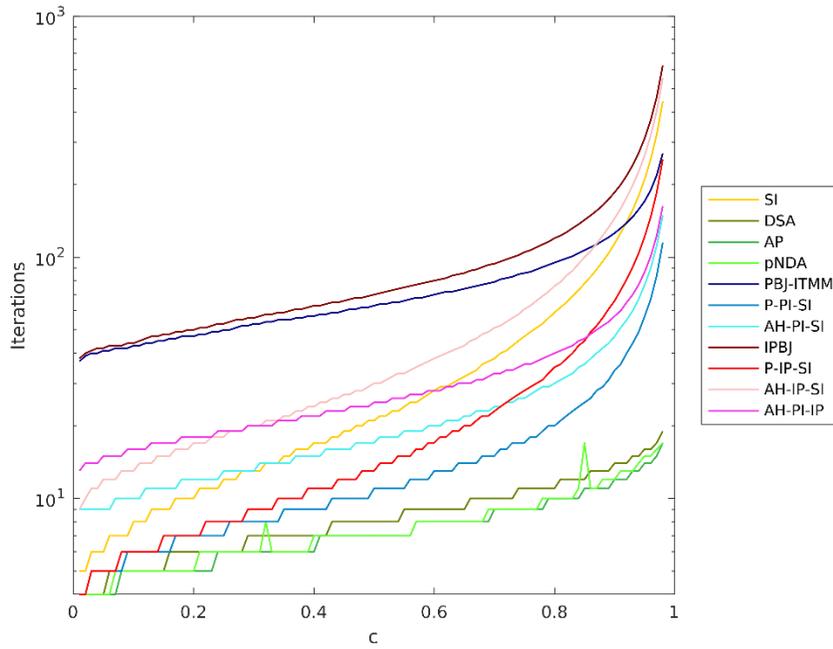


Figure 4.40: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 1$

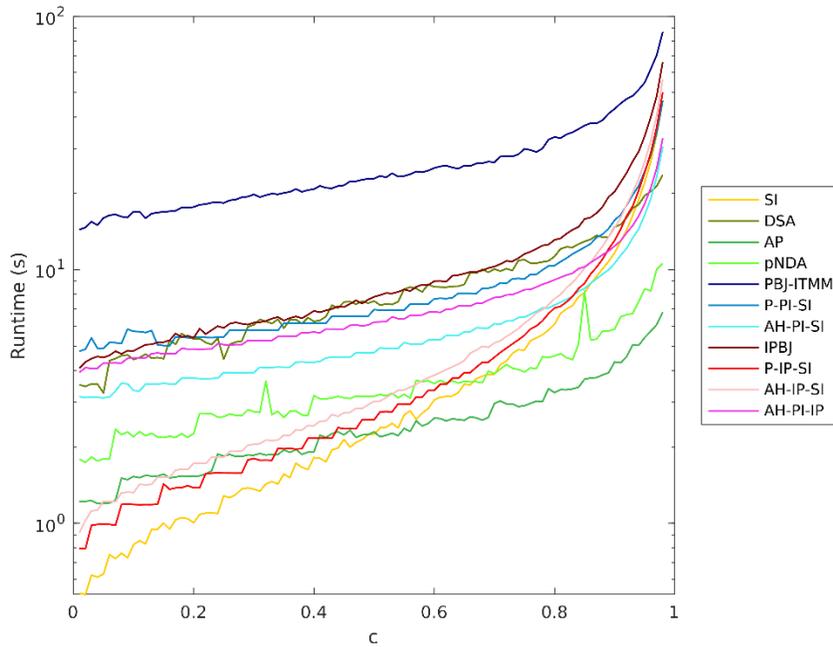


Figure 4.41: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 1$

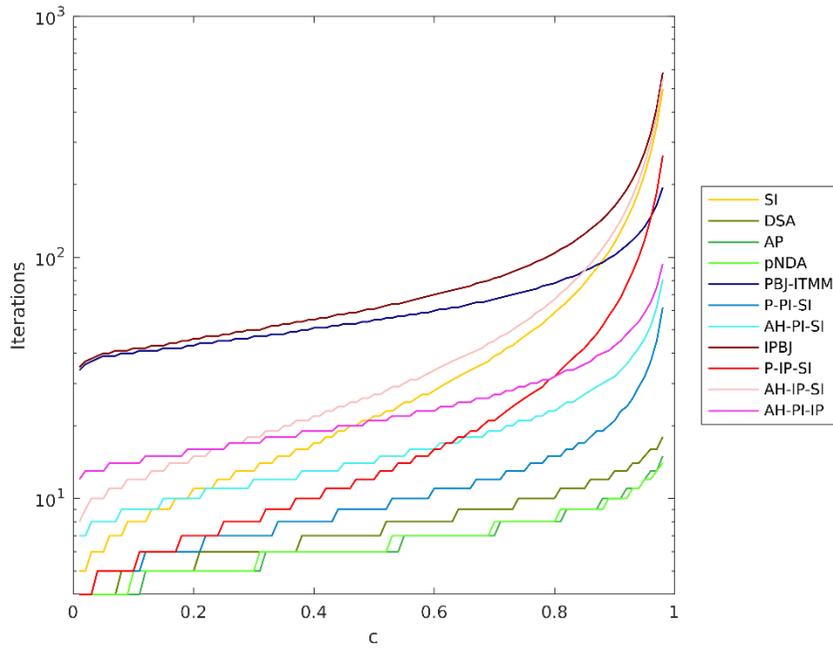


Figure 4.42: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 2.5$

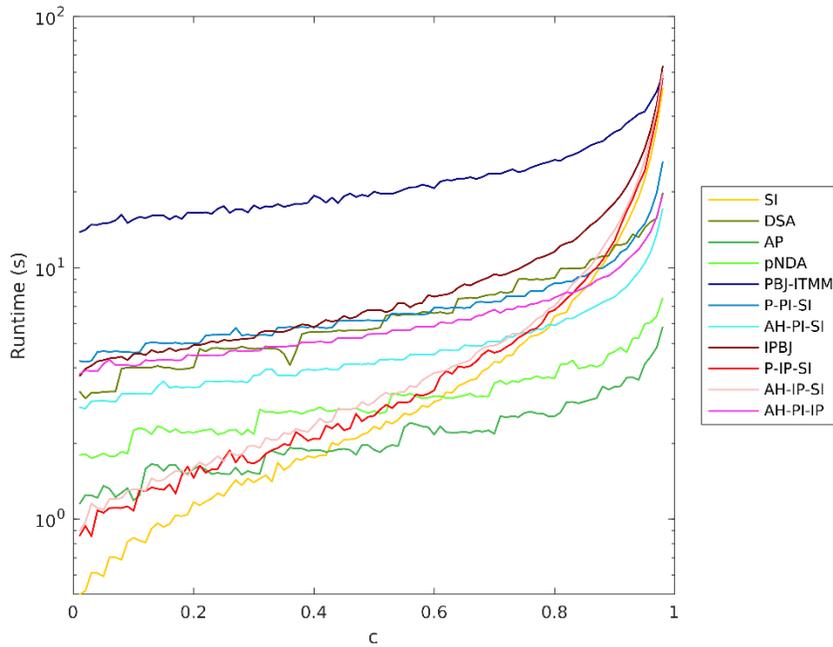


Figure 4.43: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 2.5$

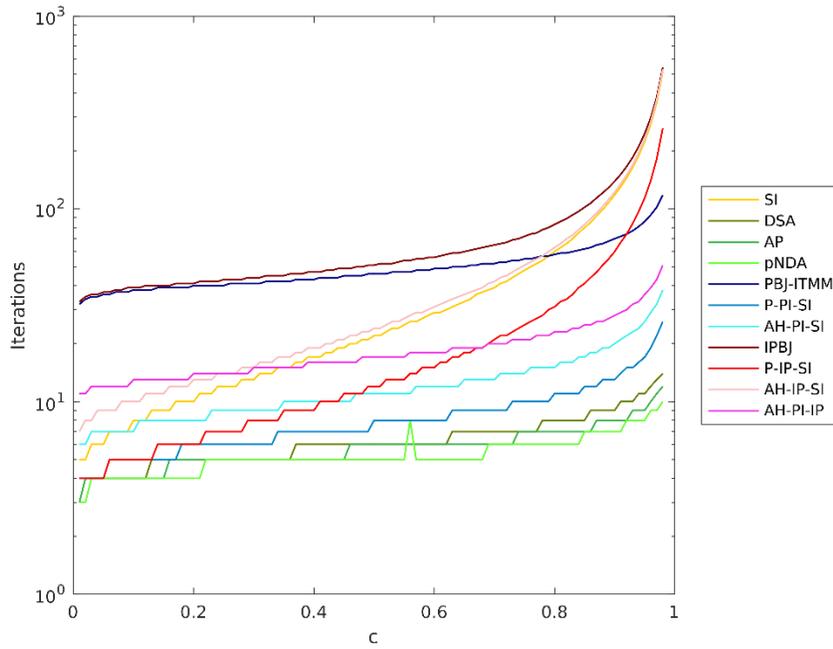


Figure 4.44: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 10$

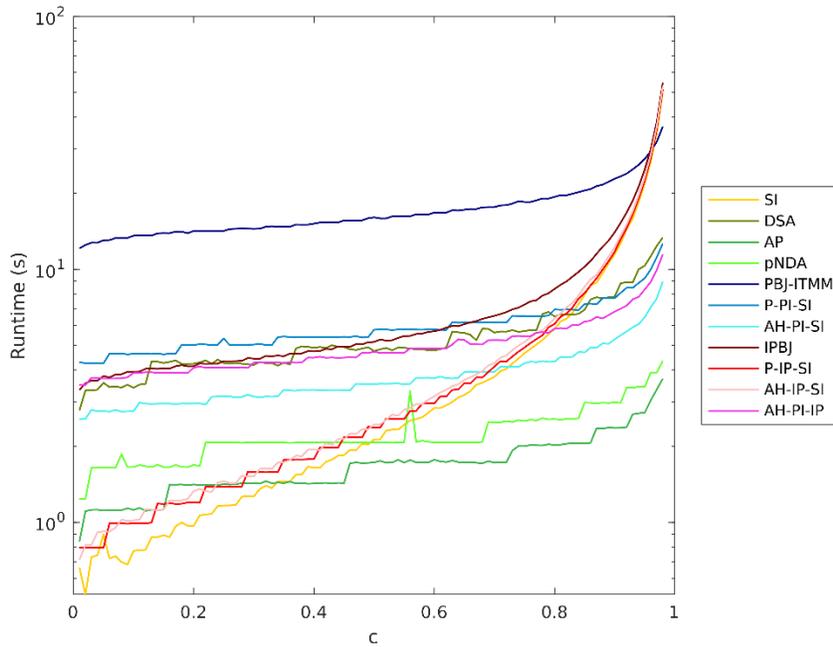


Figure 4.45: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 10$

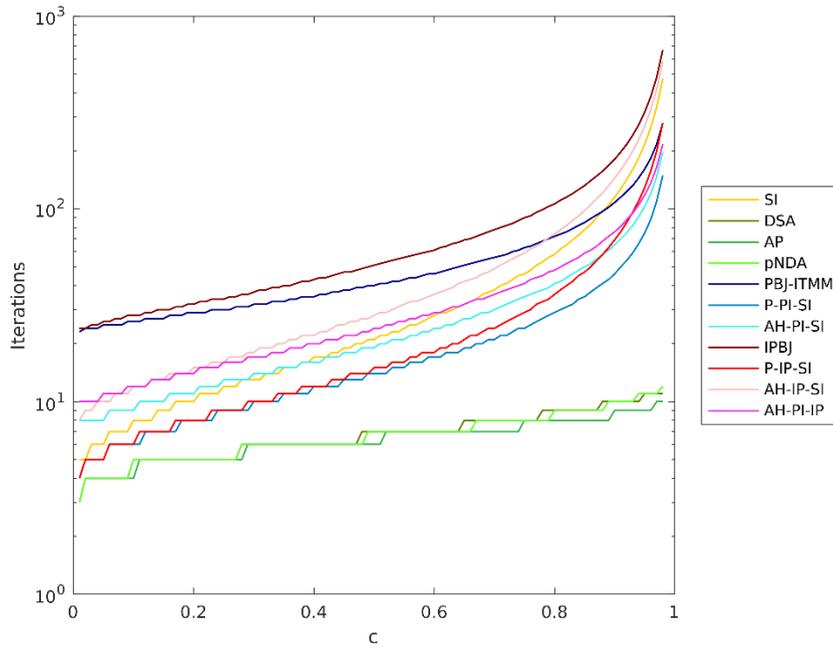


Figure 4.46: Iterations required to converge parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.1, \Sigma_{thick} \Delta u_{thick} = 1$

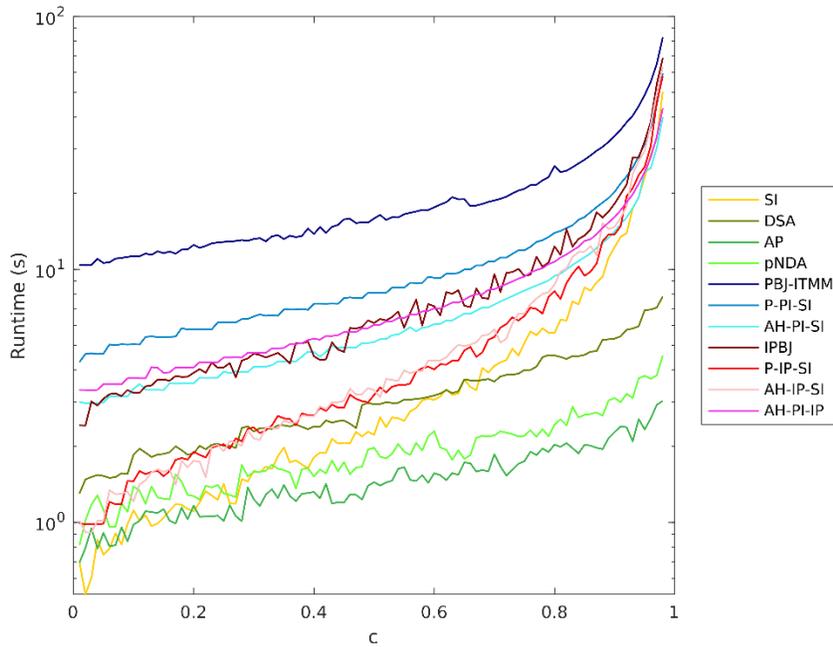


Figure 4.47: Runtime observed for parametric study problem versus scattering ratio:  
 $N_s = 16, \Sigma_{thin} \Delta u_{thin} = 0.1, \Sigma_{thick} \Delta u_{thick} = 1$

Analyzing the results of the parametric study in Figs. 4.24 - 4.47, we begin with the iterative performances of *PBJ-ITMM* and *IPBJ*. In terms of execution time, *IPBJ* is observed to converge faster than *PBJ-ITMM*, but *PBJ-ITMM*'s iterative performance degrades slower as  $c$  increases, with *PBJ-ITMM* eventually converging in less time than *IPBJ* for some cases. In terms of iteration count, however, *PBJ-ITMM* always converges in the same number of fewer iterations than *IPBJ*, as has been mathematically predicted. [4] At lower scattering ratios, *PBJ-ITMM* and *IPBJ* converge in almost the same number of iterations. Then, as  $c$  increases, the difference in required iterations between *PBJ-ITMM* and *IPBJ* becomes on the order of hundreds. The specific magnitude of this difference is case-dependent. Contrasting Figs. 4.24, 4.40, 4.42, and 4.44, it is apparent that the difference in required iterations between *PBJ-ITMM* and *IPBJ* increases as  $\Sigma_{thick}\Delta u_{thick}$  increases. This observation is attributed to the spectral properties of these two methods, displayed previously in Figs. 4.3 and 4.4. As cell size increases, for non-unity scattering ratios, the spectral radius of *PBJ-ITMM* tends towards zero, whereas the spectral radius of *IPBJ* tends towards  $c$ . Therefore, as the optical thickness of cells in the “thick” stripes increases, the difference in the spectral properties of the two methods increases, causing the observed trend. Alternatively, this trend can be interpreted physically as the diminishing effect of both methods' asynchronicity as optical thickness increases, leaving scattering as the dominant transport phenomenon, which *PBJ-ITMM* is more effective at resolving.

Contrasting Figs. 4.24, 4.34, 4.36, and 4.38, we observe that the difference in the number of required iterations between *PBJ-ITMM* and *IPBJ* is largely independent of  $\Sigma_{thin}\Delta u_{thin}$ . Since the optical thickness of cells in “thin” stripes is small, the difference in spectral radius between the two methods diminishes, with the asynchronicity of each method being the primary factor impacting iterative performance. In general, as  $\Sigma_{thin}\Delta u_{thin}$  increases, the iterative performance of both methods improves, except when  $\Sigma_{thin}\Delta u_{thin}$  is very small, in which case the increased leakage improves iterative performance. This exception is only expected to extend to problems in which the optically thin cells are on problem boundaries, however.

Contrasting Figs. 4.24, 4.26, 4.28, 4.30, and 4.32, we observe *PBJ-ITMM*'s iterative performance to improve when  $N_s$  is small. When  $N_s$  is small, the number of sub-domains that particles traverse before re-entering an optically thick region is small, thus limiting the detriment to convergence rate in optically thin cells resulting from the method's asynchronicity. In the case of  $N_s = 4$ , each stripe is only a single sub-domain wide. In this case, streaming along the  $x$ -axis

across an optically thin region can therefore be resolved with a single *PBJ* iteration. These same traits extend to *IPBJ* as well, but when  $c$  is relatively small. When  $c$  becomes large, as we have noted previously, *IPBJ* demonstrates little difference between different cases.

Next in our analysis of the parametric study results, we compare the performance of hybrid methods to their individual iterative constituents. Firstly, we have previously predicted through theoretical analysis and preliminary testing that *IPBJ* will be ineffective as the primary method in such hybrid implementations. This parametric study provides the most conclusive evidence supporting this prediction. While *P-IP-SI* and *AH-IP-SI* provide significant reductions in execution time compared to *IPBJ* when  $c$  is small, such problems are known to be rapidly convergent with unaccelerated *SI*. As  $c$  becomes large, though, the number of required iterations for *IPBJ*, *SI*, and *AH-IP-SI* become similar, with *P-IP-SI* requiring roughly half this number, due to the fact that *P-IP-SI* two-step iterative scheme. Consequently, as  $c$  approaches 1, these four methods converge in comparable execution times. The similarity in iterative performance of these four methods at high scattering ratios indicates that, in such cases, the asynchronicity of *IPBJ* is insignificant from the standpoint of convergence rate in comparison to the adverse effects of lagging the scattering source. With *IPBJ* unable to resolve local scattering, there is therefore a fundamental transport phenomenon left with no effective method for its resolution, causing hybrid methods that utilize *IPBJ* as the primary method to be ineffective in problems with even moderately large scattering ratios.

Contrarily, for hybrid methods involving *PBJ-ITMM* as the primary method, there is significant improvement over *PBJ-ITMM* in terms of both required iterations and observed execution time, even when  $c$  is large. The magnitude of a hybrid approach's improvement over *PBJ-ITMM* varies significantly between cases. Contrasting Figs. 4.24, 4.26, 4.28, 4.30, and 4.32 (and their execution time counterparts), the improvement of hybrid methods containing *PBJ-ITMM* over *PBJ-ITMM* alone is far greater when  $N_s$  is large. As  $N_s$  becomes small, however, the diminished performance improvement due to the hybrid approach is not due to a degradation of hybrid methods' performances, but rather, due to the previously noted improvement of *PBJ-ITMM*'s iterative performance when  $N_s$  becomes small. This aligns with our physical interpretation of the hybrid approach, as *PBJ* methods are capable of efficiently resolving the streaming across optically thin regions when those regions are only a few sub-domains wide. Then, comparing Figs. 4.24, 4.40, 4.42, and 4.44, hybrid methods improve in their overall performance when

$\Sigma_{thick}\Delta u_{thick}$  increases. The improvement compared to *PBJ-ITMM* is relatively constant with respect to  $\Sigma_{thick}\Delta u_{thick}$ , though, as the gained iterative performance in these hybrid methods is due to the gained iterative performance of *PBJ-ITMM* itself. However, when  $\Sigma_{thin}\Delta u_{thin}$  decreases (Figs. 4.24, 4.34, 4.36, and 4.38), the performance of hybrid methods containing *PBJ-ITMM* improves, thus increasing the performance gains over *PBJ-ITMM* alone. As  $\Sigma_{thin}\Delta u_{thin}$  decreases, in “thin” regions, long-distance streaming becomes the increasingly dominant transport phenomenon. The consequential decrease in the average number of collisions incurred by particles while traversing the thin region therefore decreases, increasing the effectiveness of *SI* in these regions. The end result of this effect is that, while *PBJ-ITMM*, in general, benefits iteratively from the thickening of all cells, hybrid methods involving *PBJ-ITMM* perform best in problems in which cell thicknesses tend towards either extreme (i.e. performance increases as optical thickness increases or decreases in thick or thin cells, respectively).

In our parametric study, we have included data for multiple hybrid methods that use *PBJ-ITMM* as the primary method. With the hybrid approach observed to be effective for accelerating *PBJ-ITMM*, we now use this data to compare the different hybrid implementations that use *PBJ-ITMM* as the primary method. Firstly, we compare the performances of *P-PI-SI* and *AH-PI-SI*. *P-PI-SI* is, as one would expect, found to converge in fewer iterations than *AH-PI-SI* in all cases. *AH-PI-SI*, however, converges in a shorter runtime than *P-PI-SI*, almost always. This is due to the fact that while the preconditioning approach provides a greater reduction in the iterative error per iteration than the asynchronous hybrid approach, it comes at the cost of increased time per iteration (since *P-PI-SI* requires a two-step iterative process, each executed globally). It is clear from this data that the reasoning behind *AH-PI-SI*'s development was correct; that in the hybrid approach, one method, as determined by the cell's optical thickness, is predominantly responsible for the convergence in a given cell-thickness regime. Additionally, the primary shortcoming of *P-PI-SI* is that it greatly reduces the degree of parallelism in our target application, due to the global *SI* sweep that must be computed in spatially serial execution. *AH-PI-SI* eliminates this global sweep in favor of sweeps over the *SI* zones, which may also be performed simultaneously with all other zones. With *AH-PI-SI* observed to converge in shorter runtimes than *P-PI-SI*, and with a higher degree of parallelism in our target application, we conclude that the asynchronous hybrid approach is a more computationally advantageous approach than the preconditioning approach, with regards to our proposed hybrid framework for iterative schemes.

With the asynchronous hybrid approach deemed preferable to preconditioning for practical application, we compare the performance of *AH-PI-SI* and *AH-PI-IP*. Recall that *AH-PI-IP* was developed in order to further reduce the penalty to the degree of parallelism of *PBJ-ITMM* compared to *AH-PI-SI*. For our parametric study, the *IPBJ* sub-domains within thin zones were of a size such that there was no reduction to the degree of parallelism. From the collected data, it is clear that comes at the cost of an increase in the required number of iterations. This is an inevitable consequence of *AH-PI-IP*, as it imposes asynchronicities that were not present with *AH-PI-SI*. In many cases though, the adverse impact on convergence rate is small, while reclaiming the full degree of parallelism of *PBJ-ITMM* in our target application. In our parametric study, two parameters are seen to impact the difference in convergence rate between *AH-PI-SI* and *AH-PI-IP*, namely  $N_s$  and  $\Sigma_{thin}\Delta u_{thin}$ . As  $N_s$  increases, the number of iterations (and execution time) of *AH-PI-IP* compared to *AH-PI-SI* increases as well. As  $N_s$  increases, with *AH-PI-IP*, the number of *IPBJ* sub-domains per thin region increases, thus increasing the number of iterations required to model the long-distance streaming prevalent in such regions. Similarly, when  $\Sigma_{thin}\Delta u_{thin}$  decreases, the average distance traveled within thin regions increases, increasing the difference in performance between *AH-PI-SI* and *AH-PI-IP*. The result is a trade-off between the two methods between the degree of parallelism and iterative effectiveness.

Recall that for our *AH-PI-IP* implementation, we consider *IPBJ* sub-domains of a size such that the time to sweep the sub-domain over a single angle does not exceed the *PBJ-ITMM* sub-domain solution time. Therefore, our comparison of *AH-PI-IP* to *AH-PI-SI* provides the two extremes in this case, with either the full “thin” zone being swept synchronously, or with *IPBJ* sub-domains as small as would be possibly advantageous for such an implementation. In practice, however, *IPBJ* sub-domains could exceed this size limit, increasing iterative effectiveness at the cost of a reduced degree of parallelism, with *PBJ-ITMM* sub-domains completing their iterative solution, then having to wait for *IPBJ* sub-domains to complete their sweeps before exchanging angular fluxes. For practical implementation, the optimal *IPBJ* sub-domain size will likely be between the two extremes, with the ideal size dependent on problem parameters and HPC environment. For problems where the asynchronous hybrid approach is applicable, namely problems with at least one large, contiguous region of optically thin cells, the larger this region is and the thinner the cells it contains are, the larger the difference in convergence rate between *AH-PI-SI* and *AH-PI-IP*. As the number of cells in this thin region increases, however, this difference

is supplemented by the fact that *AH-PI-SI*'s penalty to the degree of parallelism also increases in these cases. In general, we may conclude that the ideal *IPBJ* sub-domain size for *AH-PI-IP* will likely increase as the thin region become optically thinner or as the number of constituent cells increases.

For the HPC environment's impact on optimal *IPBJ* sub-domain size, as will be discussed in subsequent sections and chapters, in parallel operation, each iteration incurs a communication time penalty in addition to the computation time. The effect of this communication time on the ideal *IPBJ* sub-domain size in the *AH-PI-IP* method is less obvious. In HPC environments and computer codes where the communication time is heavily dominant compared to the computation time, the reduced iteration count of *AH-PI-SI* (or *AH-PI-IP* with increased *IPBJ* sub-domain size) is preferable, as this will decrease the total number of communication steps. However, this also increases the per-iteration communication requirements of the processor solving an *IPBJ* sub-domain (or *SI* zone). These competing effects render the ideal *IPBJ* sub-domain size for *AH-PI-IP* HPC-specific and not inferable without studying the method on the target platform. Our parametric study, however, demonstrates both *AH-PI-SI* and *AH-PI-IP* to dramatically decrease the number of iterations and runtime compared to *PBJ-ITMM*, with little to no penalty to the potential degree of parallelism.

The final analysis of the data collected through our parametric study pertains to the three traditional acceleration methods (*DSA*, *AP*, and *pNDA*) and the comparison of their performances to *PBJ* and hybrid methods. From the presented cases it is clear that, in terms of required iterations, all three acceleration methods perform very similarly, with *DSA* typically underperforming the other two by a very small margin. *AP* and *pNDA* are extremely similar in iteration counts, with the method that performs slightly better depending on the specific case. In terms of runtime, however, *AP* is almost universally the fastest of the three methods, with *pNDA* requiring slightly longer runtimes, and *DSA* requiring significantly longer runtimes still. The difference in runtimes so greatly exceeding the difference in required iterations is a consequence of the different matrix solution times for the various acceleration methods. *DSA* is based on cell-edge quantities, resulting in a matrix of rank  $2N_xN_y + N_x + N_y$ , making it larger than the matrices for *AP* and *pNDA*, which are both of rank  $N_xN_y + 2(N_x + N_y)$  due to the fact that they are based on cell-centered quantities. Thus, the *DSA* matrix consumes a longer execution time to solve, causing each iteration to take longer than when *AP* or *pNDA* are used. While the *AP* and *pNDA* matrices are of the same size,

*AP* is a linear method and therefore, the associated matrix need only be constructed and *LU*-decomposed only once, as its elements do not change between iterations. *pNDA*, however, is nonlinear, meaning that the matrix elements are themselves dependent on the solution of the transport sweep. These matrix elements consequently change in each iteration, requiring repeated reconstruction of the *pNDA* matrix. This results in the slightly longer execution times observed for *pNDA*, despite having very similar iteration counts.

There is a feature observed with *pNDA* not present with any other method studied. While all execution time observation trends are not smooth due to the uncertainty associated with timing runs, *pNDA* is seen to have multiple pronounced spikes in runtime that are not due to these runtime uncertainties, but rather, actual spikes in the required number of iterations. Upon investigation, it was discovered that these spikes were due, not to a degradation of *pNDA*'s iterative performance in these cases, but due to a combination of the closure between the high-order and low-order problems on which *NDA* methods are based, and the *BiCGSTAB* iterations HAT-2C uses to solve the *pNDA* matrix. When the low order problem is solved iteratively, its solution contains iterative error. HAT-2C converges the *BiCGSTAB* iterations for the low order problem to the same stopping criterion as designated for the transport solution. *DSA* and *AP* operate on iterative change, rather than the scalar flux, and therefore cease to impact the iterative solution as convergence is approached. *pNDA*, on the other hand, produces the updated scalar fluxes directly, relying on the low order problem's closure with the high order problem to converge the problem to the same solution as the unaccelerated method. It is therefore possible for the iterative error in the low order problem to delay the determination of convergence of the iterative sequence. This conjecture was tested by manually tightening the stopping criterion of the *BiCGSTAB* iterations. When this stopping criterion was tightened, the aforementioned spikes in *pNDA*'s number of required iterations were eliminated, thus confirming that they were a consequence of the iterative error in the low order solution, rather than a degradation in *pNDA*'s spectral properties. For our results, we display those produced when *pNDA*'s low order solution is converged only to the designated stopping criterion, as tightening of this would increase the execution time for *pNDA* in all cases, whereas these spikes are only present in a small fraction of cases.

*AP* is found to be the best performing acceleration method of the three considered, almost universally, and is therefore the standard against which we compare *PBJ* and hybrid methods. With a few exceptions, *AP* requires fewer iterations and shorter runtime to converge than *PBJ-ITMM* or

any hybrid method in which *PBJ-ITMM* is used. This is expected, as *AP* is an acceleration method intended for serial execution in Cartesian grids. It is not readily extendible to parallel execution, much less parallel execution on unstructured grids. While *AP* is consistently found to converge in fewer iterations and shorter execution times than any developed hybrid method, *AH-PI-IP*, and by extension *AH-PI-SI*, typically execute in times within a factor of ten greater than *AP* in most cases. With *PBJ*-type methods being extendible to the massively parallel regime, we view this result as establishing the case for adopting hybrid iterative schemes in  $S_N$  parallel solution algorithms on unstructured grids.

From our parametric study, we have reached 5 primary conclusions:

- (1) The asynchronous hybrid approach is effective for accelerating *PBJ-ITMM* in problems containing both optically thick and thin cell
- (2) This acceleration is maximized in problems with large regions of very thin cells
- (3) Both *AH-PI-SI* and *AH-PI-IP* significantly accelerate *PBJ-ITMM*, but *AH-PI-IP*'s acceleration decreases when a region of thin cells becomes large enough, indicating that the *IPBJ* sub-domain size when using *AH-PI-IP* will likely be problem and HPC environment dependent
- (4) The hybrid approach is far less effective with *IPBJ* as the primary method due to *IPBJ*'s inability to resolve scattering events in a single iteration
- (5) While traditional acceleration methods are consistently found to converge in fewer iterations and shorter execution times than our hybrid approach, *AH-PI-IP* and *AH-PI-SI* typically require execution times within a factor of ten longer than these methods, which are not readily extendible to our target applications.

## 4.7: Parametric Study with Extreme Scattering Ratios

The parametric study detailed in the previous section spans the range  $c \in [0.01, 0.98]$ . This parametric study was replicated for the  $c$  values,  $c = \{0.99, 0.999, 0.9999, 0.99999, 1.0\}$ , the results and analysis of which is the subject of this section. The results of the parametric study for these extreme scattering ratios are separated from the rest for two reasons. Firstly, our development of hybrid methods was targeted at alleviating the iterative slowdown incurred by *PBJ*-type methods in problems containing optically thin cells. This development was not intended or expected to provide iterative robustness as the scattering ratio approaches unity. The parametric study with extreme scattering ratios is therefore intended to examine and quantify this issue, rather than address its resolution. Secondly, in this region where  $c$  approaches unity, the number of iterations for many methods increases dramatically, with many exceeding our maximum number of iterations. The data for the parametric study with extreme scattering ratios is therefore better suited to be displayed in tables, rather than graphs. Below in Tables. 4.6 - 4.17, we provide results for selected cases. The full suite of test results is available in Appendix B. For instructions on interpreting the color coded table captions, refer to the previous section. In these tables, N/C indicates a method did not converge in the maximum number of iterations, 500 for *P-PI-SI* and *P-IP-SI*, 1000 for all others. N/A is a designation reserved for *DSA* when  $c = 1$ . Referring to Eq. (2.12), it is clear that the *DSA* equations in this form incur a divide-by-zero error when  $c = 1$ .

Table 4.6: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$  (Base Case)

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	935 / 89.62	N/C	N/C	N/C	N/C
<i>DSA</i>	18 / 19.48	26 / 32.47	32 / 45.16	34 / 47.70	N/A
<i>AP</i>	15 / 5.35	23 / 9.33	29 / 13.40	31 / 14.40	31 / 14.42
<i>pNDA</i>	13 / 6.39	19 / 11.33	22 / 14.22	23 / 14.88	23 / 14.95
<i>PBJ-ITMM</i>	200 / 59.93	590 / 171.69	N/C	N/C	N/C
<i>P-PI-SI</i>	64 / 27.63	331 / 128.88	N/C	N/C	N/C
<i>AH-PI-SI</i>	83 / 17.81	380 / 75.88	774 / 152.69	872 / 171.91	885 / 174.44
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	482 / 93.82	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	982 / 97.81	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	96 / 19.99	405 / 79.81	820 / 160.30	923 / 180.25	936 / 182.13

Table 4.7: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	917 / 97.09	N/C	N/C	N/C	N/C
<i>DSA</i>	26 / 25.13	49 / 64.33	88 / 125.19	98 / 142.78	N/A
<i>AP</i>	23 / 7.74	47 / 20.74	84 / 41.48	94 / 45.79	95 / 45.00
<i>pNDA</i>	21 / 10.88	41 / 35.12	65 / 65.47	68 / 67.97	69 / 69.04
<i>PBJ-ITMM</i>	136 / 45.61	735 / 235.16	N/C	N/C	N/C
<i>P-PI-SI</i>	71 / 32.29	398 / 163.23	N/C	N/C	N/C
<i>AH-PI-SI</i>	105 / 22.56	581 / 117.24	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	494 / 105.14	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	980 / 99.30	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	109 / 23.43	593 / 118.92	N/C	N/C	N/C

Table 4.8: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	920 / 95.20	N/C	N/C	N/C	N/C
<i>DSA</i>	20 / 21.35	36 / 50.42	53 / 85.32	57 / 91.83	N/A
<i>AP</i>	17 / 6.29	34 / 16.51	49 / 25.97	53 / 27.31	54 / 27.61
<i>pNDA</i>	15 / 7.78	29 / 22.89	36 / 31.49	38 / 33.66	38 / 33.22
<i>PBJ-ITMM</i>	154 / 49.56	665 / 203.52	N/C	N/C	N/C
<i>P-PI-SI</i>	68 / 31.02	361 / 152.50	N/C	N/C	N/C
<i>AH-PI-SI</i>	91 / 20.58	450 / 94.99	980 / 206.53	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	475 / 101.67	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	973 / 98.27	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	96 / 21.68	468 / 99.99	N/C	N/C	N/C

Table 4.9: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	931 / 97.60	N/C	N/C	N/C	N/C
<i>DSA</i>	17 / 21.19	22 / 27.97	24 / 32.16	25 / 33.22	N/A
<i>AP</i>	14 / 5.73	19 / 7.64	21 / 9.09	22 / 9.76	22 / 9.49
<i>pNDA</i>	11 / 5.89	16 / 8.77	17 / 9.67	17 / 9.43	17 / 9.02
<i>PBJ-ITMM</i>	259 / 76.84	531 / 154.57	N/C	N/C	N/C
<i>P-PI-SI</i>	59 / 25.24	335 / 130.42	N/C	N/C	N/C
<i>AH-PI-SI</i>	76 / 16.14	398 / 77.77	976 / 188.63	N/C	N/C
<i>IPBJ</i>	992 / 98.50	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	480 / 93.30	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	975 / 95.50	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	107 / 22.13	409 / 80.30	993 / 193.28	N/C	N/C

Table 4.10: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	921 / 88.29	N/C	N/C	N/C	N/C
<i>DSA</i>	16 / 16.72	18 / 18.81	18 / 19.31	19 / 20.66	N/A
<i>AP</i>	12 / 4.25	14 / 4.93	15 / 5.72	15 / 5.68	15 / 5.82
<i>pNDA</i>	10 / 4.61	12 / 5.60	13 / 6.18	13 / 6.43	13 / 6.37
<i>PBJ-ITMM</i>	193 / 58.23	471 / 137.40	N/C	N/C	N/C
<i>P-PI-SI</i>	57 / 24.50	357 / 139.48	N/C	N/C	N/C
<i>AH-PI-SI</i>	72 / 15.11	470 / 91.08	N/C	N/C	N/C
<i>IPBJ</i>	965 / 95.92	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	474 / 92.20	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	963 / 94.08	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	77 / 16.25	470 / 91.80	N/C	N/C	N/C

Table 4.11: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	930 / 100.57	N/C	N/C	N/C	N/C
<i>DSA</i>	27 / 41.11	43 / 77.08	54 / 101.46	56 / 105.33	N/A
<i>AP</i>	21 / 9.67	35 / 18.74	45 / 25.93	48 / 27.07	48 / 26.98
<i>pNDA</i>	17 / 10.38	25 / 18.62	29 / 21.79	29 / 22.30	29 / 21.85
<i>PBJ-ITMM</i>	195 / 58.42	563 / 164.13	N/C	N/C	N/C
<i>P-PI-SI</i>	61 / 26.29	316 / 123.12	N/C	N/C	N/C
<i>AH-PI-SI</i>	79 / 16.86	362 / 72.35	716 / 141.30	801 / 158.37	812 / 160.42
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	479 / 93.10	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	977 / 97.28	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	90 / 18.87	384 / 76.59	756 / 147.79	845 / 165.02	856 / 167.13

Table 4.12: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	955 / 97.62	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 8.59	13 / 12.15	13 / 13.93	13 / 13.97	N/A
<i>AP</i>	10 / 2.43	11 / 3.60	11 / 4.03	11 / 4.07	11 / 4.20
<i>pNDA</i>	10 / 3.39	11 / 4.86	12 / 5.83	12 / 5.94	12 / 6.03
<i>PBJ-ITMM</i>	222 / 72.28	743 / 235.22	N/C	N/C	N/C
<i>P-PI-SI</i>	102 / 42.07	465 / 183.74	N/C	N/C	N/C
<i>AH-PI-SI</i>	140 / 29.08	563 / 111.54	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	497 / 96.55	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	162 / 32.86	608 / 119.17	N/C	N/C	N/C

Table 4.13: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	964 / 103.48	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 3.51	8 / 5.16	9 / 7.53	9 / 8.02	N/A
<i>AP</i>	7 / 1.37	7 / 1.81	7 / 2.22	7 / 2.40	7 / 2.29
<i>pNDA</i>	10 / 2.53	10 / 3.45	10 / 3.94	11 / 4.78	11 / 5.00
<i>PBJ-ITMM</i>	211 / 72.12	846 / 269.40	N/C	N/C	N/C
<i>P-PI-SI</i>	160 / 69.04	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	382 / 80.87	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	403 / 85.47	N/C	N/C	N/C	N/C

Table 4.14: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	722 / 77.96	N/C	N/C	N/C	N/C
<i>DSA</i>	21 / 29.14	28 / 45.12	30 / 47.46	30 / 49.30	N/A
<i>AP</i>	19 / 8.58	26 / 13.69	29 / 15.11	29 / 14.79	29 / 14.88
<i>pNDA</i>	20 / 12.56	25 / 17.47	26 / 18.61	26 / 19.39	26 / 18.93
<i>PBJ-ITMM</i>	390 / 121.99	900 / 283.49	N/C	N/C	N/C
<i>P-PI-SI</i>	188 / 84.10	493 / 203.68	N/C	N/C	N/C
<i>AH-PI-SI</i>	238 / 51.22	604 / 127.48	728 / 150.79	744 / 148.69	745 / 147.09
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	414 / 89.83	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	899 / 89.35	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	256 / 51.14	643 / 125.86	775 / 152.91	791 / 154.38	793 / 154.95

Table 4.15: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	878 / 91.81	N/C	N/C	N/C	N/C
<i>DSA</i>	20 / 24.59	28 / 41.09	32 / 49.81	33 / 50.51	N/A
<i>AP</i>	17 / 6.91	25 / 12.31	29 / 14.55	30 / 15.09	30 / 14.78
<i>pNDA</i>	15 / 8.40	22 / 14.53	24 / 16.25	25 / 17.34	25 / 17.18
<i>PBJ-ITMM</i>	264 / 83.94	772 / 238.55	N/C	N/C	N/C
<i>P-PI-SI</i>	106 / 46.12	447 / 186.15	N/C	N/C	N/C
<i>AH-PI-SI</i>	133 / 29.33	516 / 108.36	805 / 168.21	856 / 179.15	862 / 180.15
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	466 / 98.61	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	965 / 102.52	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	147 / 33.01	548 / 118.57	853 / 184.26	906 / 197.05	912 / 196.36

Table 4.16: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ ,  $\Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	963 / 101.54	N/C	N/C	N/C	N/C
<i>DSA</i>	16 / 16.97	23 / 30.05	31 / 48.21	34 / 54.02	N/A
<i>AP</i>	14 / 4.77	21 / 9.01	29 / 14.51	31 / 15.67	32 / 16.06
<i>pNDA</i>	11 / 5.40	16 / 10.12	20 / 13.35	21 / 14.95	21 / 14.75
<i>PBJ-ITMM</i>	153 / 49.89	423 / 132.91	N/C	N/C	N/C
<i>P-PI-SI</i>	39 / 18.91	217 / 89.58	N/C	N/C	N/C
<i>AH-PI-SI</i>	54 / 12.05	252 / 50.87	692 / 138.55	864 / 170.99	889 / 176.55
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	489 / 95.76	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	988 / 98.94	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	68 / 14.61	272 / 54.27	734 / 144.04	914 / 179.22	941 / 186.04

Table 4.17: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	806 / 77.27	N/C	N/C	N/C	N/C
<i>DSA</i>	11 / 8.37	12 / 11.32	12 / 11.98	12 / 12.16	N/A
<i>AP</i>	10 / 2.73	10 / 3.25	11 / 3.76	11 / 3.81	11 / 3.73
<i>pNDA</i>	12 / 4.37	13 / 5.63	13 / 5.95	13 / 5.80	13 / 5.98
<i>PBJ-ITMM</i>	432 / 126.18	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	248 / 97.26	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	322 / 64.44	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	476 / 92.56	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	993 / 98.66	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	349 / 68.98	N/C	N/C	N/C	N/C

From the parametric study with extreme scattering ratios, we analyze the performance of *PBJ* hybrid methods as the scattering ratio approaches unity. From these results it is clear that no *PBJ* or hybrid method studied is iteratively robust with respect to scattering ratio. This result is expected, with the hybrid approach designed to improve iterative performance in optically thin cells, not large scattering ratios, and all theoretical analysis indicating a lack of robustness in this regime. In the original parametric study, while both *PBJ-ITMM* and *IPBJ* experienced an increase in iteration counts as  $c$  increases, the increase was far sharper with *IPBJ*. It is clear that, while *PBJ-ITMM* is less sensitive to  $c$  than *IPBJ*, when  $c$  becomes very large ( $>0.98$ ), *PBJ-ITMM*'s iterative performance also begins to degrade dramatically. In our extreme scattering ratio parametric study, however, this approach is observed to improve iterative performance as  $c$  approaches unity for many cases. In some cases, *AH-PI-SI* (and potentially *AH-PI-IP*) is (are) seen to converge in fewer than 1,000 iterations with unity scattering ratio.

Interestingly, the effects of some problem parameters on the convergence rates of *PBJ-ITMM* and hybrid methods containing it reverse trends when the scattering ratio approaches 1. Contrasting Tables 4.6, 4.11, 4.12, and 4.13, as  $\Sigma_{thin}\Delta u_{thin}$  increases, the performance of *PBJ-ITMM* becomes worse, contrary to the results of the previous parametric study and predictions pertaining to *PBJ-ITMM*'s iterative behavior. To explain this phenomenon, we refer to Fig. 4.18. This graph demonstrates that *PBJ-ITMM*'s spectral radius in the infinite medium analysis is dependent almost entirely on  $\Sigma_a$ . As the scattering ratio approaches unity,  $\Sigma_a$  approaches zero. Thus, as the scattering ratio approaches unity, the benefit to *PBJ-ITMM*'s iterative performance from increased optical thickness disappears, with the lack of absorption resulting in particles crossing many sub-domain boundaries, regardless of how optically thick those sub-domains may be. Therefore, when  $\Sigma_a$  approaches zero as a consequence of  $c$  approaching unity, the benefit of increased leakage outweighs the benefit of slightly increased absorption, and iterative performance actually improves when  $\Sigma_{thin}\Delta u_{thin}$  decreases.

With  $\Sigma_{thick}\Delta u_{thick}$  being larger than  $\Sigma_{thin}\Delta u_{thin}$ , this phenomenon occurs at a larger scattering ratio, since  $\Sigma_a$  will tend towards zero with increasing  $c$  slower due to the increased  $\Sigma_t$ . This phenomenon is therefore not observed with *PBJ-ITMM* in our study, as the method exceeds 1,000 iterations before the scattering ratio become large enough. However, for the cases where we change  $\Sigma_{thick}\Delta u_{thick}$  (Tables 4.6, 4.14, 4.15, and 4.16), both *AH-PI-SI* and *AH-PI-IP* converge for all scattering ratios. In the original parametric study, both of these methods converge faster when

$\Sigma_{thick}\Delta u_{thick}$  is larger, which is a direct consequence of *PBJ-ITMM* converging faster in such circumstances. Contrasting the aforementioned tables, though, both *AH-PI-SI* and *AH-IP-IP* converge faster when  $\Sigma_{thick}\Delta u_{thick}$  is smaller. While not directly observed due to iterations exceeding 1,000, all evidence suggests that *PBJ-ITMM* would exhibit the same phenomenon, were there to be no limit on the number of iterations.

The convergence rates of *PBJ*-type methods and *SI* are determined by the number of particle-terminating events (crossing sub-domain boundaries for *PBJ-ITMM*, scattering for *SI*, either for *IPBJ*) that particles undergo on average, before loss via absorption or leakage. From the parametric study with extreme scattering ratios, we learn that, as  $c$  becomes large, absorption becomes negligible and convergence becomes based entirely on how many particle-terminating events particles undergo before exiting the problem domain. While no *PBJ*-type or developed hybrid methods are able to provide robust iterative convergence in this regime for a general case, *AH-PI-SI* and *AH-PI-IP* provide significantly improved performance over *PBJ-ITMM* and *SI*, converging in under 1,000 iterations for many cases, even when  $c=1$ .

## 4.8: Test Problems Containing Void Regions

With theoretical analysis and numerical testing on simple test problems completed for our hybrid approach, we perform numerical testing using HAT-2C on previously developed realistic test problems for deterministic neutron transport. We begin with three problems from a benchmark test suite of problems containing void regions. [67] Initially intended as a solution accuracy benchmark for problems subject to severe ray effects, the large void regions in these problems provide excellent test configurations for our hybrid approach. These problems were originally proposed in 3-dimensional geometry. Since HAT-2C is a 2-D code, the geometries are projected onto a 2-D plane. These problems are solved with HAT-2C using the 11 previously discussed iterative methods (*SI*, *DSA*, *AP*, *pNDA*, *PBJ-ITMM*, *P-PI-SI*, *AH-PI-SI*, *IPBJ*, *P-IP-SI*, *AH-IP-SI*, and *AH-PI-IP*), for a variety of scattering ratios and optical thicknesses of the non-void regions. The reported values are required iteration counts and measured execution times to converge the problems to a relative stopping criterion of  $10^{-6}$ . The problem configurations, performance results and their discussion for these three test problems are covered in the remainder of this section.

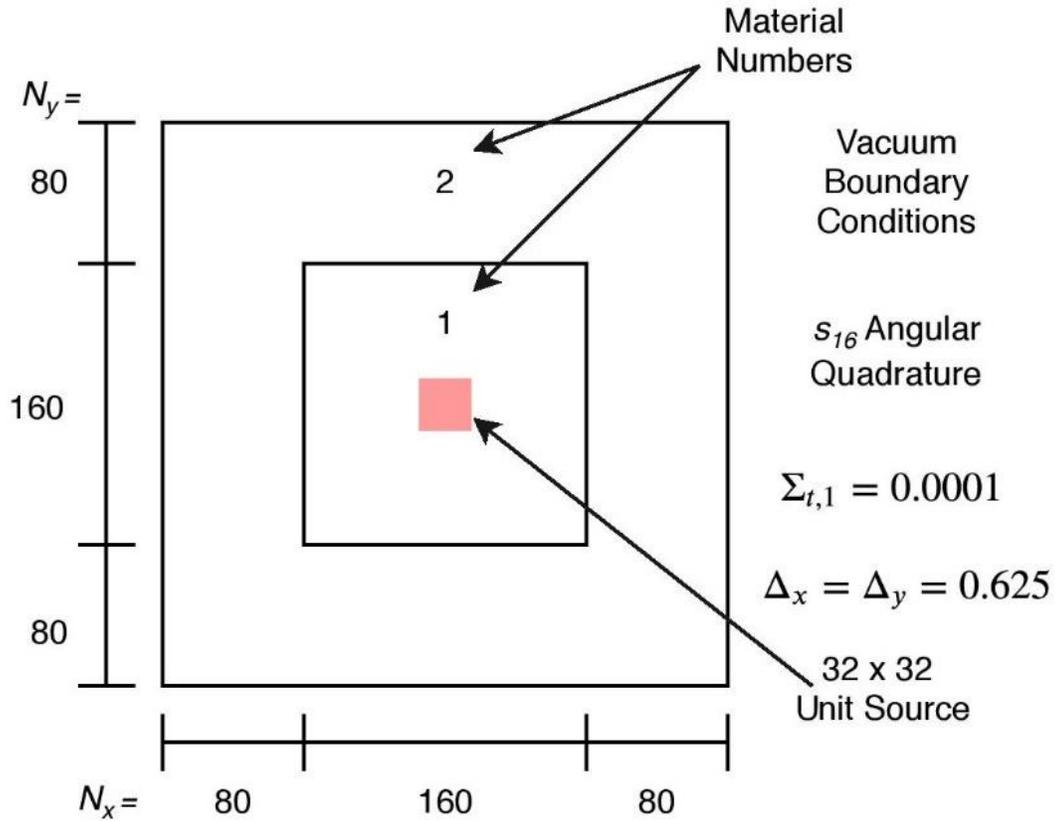


Figure 4.48: Geometry, mesh, and problem parameters for the Center Void Problem

Table 4.18: Iterations required to converge Center Void Problem

$c =$	0.5			0.9			0.98		
$\Sigma_{t,2} =$	0.1	1.0	5.0	0.1	1.0	5.0	0.1	1.0	5.0
<i>SI</i>	24	65	172	100	324	807	234	974	N/C
<i>DSA</i>	8	26	N/C	13	38	72	15	19	50
<i>AP</i>	8	28	58	13	43	72	15	21	69
<i>pNDA</i>	N/C	N/C	N/C	13	N/C	N/C	14	N/C	N/C
<i>PBJ-ITMM</i>	325	463	359	750	641	470	N/C	N/C	920
<i>P-PI-SI</i>	22	36	44	86	121	93	200	336	200
<i>AH-PI-SI</i>	127	64	54	376	160	104	821	432	224
<i>IPBJ</i>	334	474	418	807	840	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	22	43	95	88	198	424	205	N/C	N/C
<i>AH-IP-SI</i>	140	115	207	451	462	885	1000	N/C	N/C
<i>AH-PI-IP</i>	150	138	109	427	234	165	921	550	335

Table 4.19: Runtime observed in seconds to converge Center Void Problem

$c =$	0.5			0.9			0.98		
	$\Sigma_{t,2} =$	0.1	1.0	5.0	0.1	1.0	5.0	0.1	1.0
<i>SI</i>	165	480	1179	717	2219	5732	1606	6668	N/C
<i>DSA</i>	78	226	N/C	139	343	574	159	195	446
<i>AP</i>	159	418	587	297	741	951	341	454	1166
<i>pNDA</i>	N/C	N/C	N/C	155	N/C	N/C	166	N/C	N/C
<i>PBJ-ITMM</i>	3458	5012	3829	8552	6724	5022	N/C	N/C	9929
<i>P-PI-SI</i>	477	779	854	1691	2179	1803	3574	5868	3550
<i>AH-PI-SI</i>	1214	730	562	3744	1516	1092	7472	4258	2090
<i>IPBJ</i>	1282	1822	1618	3328	3228	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	236	482	1016	996	2116	4774	2193	N/C	N/C
<i>AH-IP-SI</i>	575	513	848	1874	1897	3967	4132	N/C	N/C
<i>AH-PI-IP</i>	1387	1394	1028	3930	2122	1540	8130	4896	3021

The center void problem's configuration is depicted in Fig. 4.48, and the number of iterations to achieve the stopping criterion and the execution time for all considered iterative schemes and various combinations of  $c$  and are reported in Tables 4.18 and 4.19, respectively. From the results, it is clear that this configuration is exceedingly difficult for *PBJ*-type methods to converge. This is an expected result, as the void region is fully surrounded by the non-void region, with the source in the center of the void. Consequently, in addition to the asynchronicity of *PBJ* causing iterative slowdown, if the non-void region is heavily attenuating, this problem will have very little leakage. As a result, many *PBJ-ITMM* and *IPBJ* cases fail to converge in 1000 iterations. In general, the trends observed in the parametric study are observed in this test problem. *P-PI-SI* greatly decreases the number of iterations required compared to *PBJ-ITMM*, with *AH-PI-SI* generally incurring only a modest increase in iteration count. Furthermore, *AH-PI-IP* generally requires a small enough number of iterations greater than *AH-PI-SI* to make it justifiable, especially considering that the large, continuous void region in this problem would likely cause a significant penalty to the degree of parallelism when using *AH-PI-SI*. Also, consistent with expectations, hybrid methods using *IPBJ* as the primary method do not provide sufficient acceleration as those using *PBJ-ITMM*, nor is the acceleration as consistent across cases.

The exception to the *AH* approach's performance comes when  $\Sigma_{t,2} = 0.1$ . Under this condition, *AH-PI-SI* is observed to require far more iterations for convergence than *P-PI-SI*, indicating that *SI* contributes significantly to convergence of the non-void region. This is also supported by the observation that *PBJ-ITMM* and *IPBJ* require comparable numbers of iterations under this condition, indicating that the ability of *PBJ-ITMM* to resolve scattering in the non-void region is of little consequence relative to the effect of *PBJ*'s asynchronicity. This effect was observed in the parametric study, with the hybrid method providing less acceleration when the optical thickness of the thick regions was decreased, even though it was never observed to this degree as the optical thickness of the thick regions was never made this small. Even with *AH-PI-SI* and *AH-PI-IP*, under this condition, requiring far more iterations than *P-PI-SI*, though, both hybrid methods provide significant acceleration of *PBJ-ITMM*, while maintaining most or all of its degree of parallelism.

From the center void problem, the primary conclusions are that the trends observed in the parametric study extend to this configuration that is closer to realistic applications than the periodic vertical interface test problem, and that *AH-PI-IP* provides substantial acceleration to *PBJ-ITMM* while not reducing its degree of parallelism.

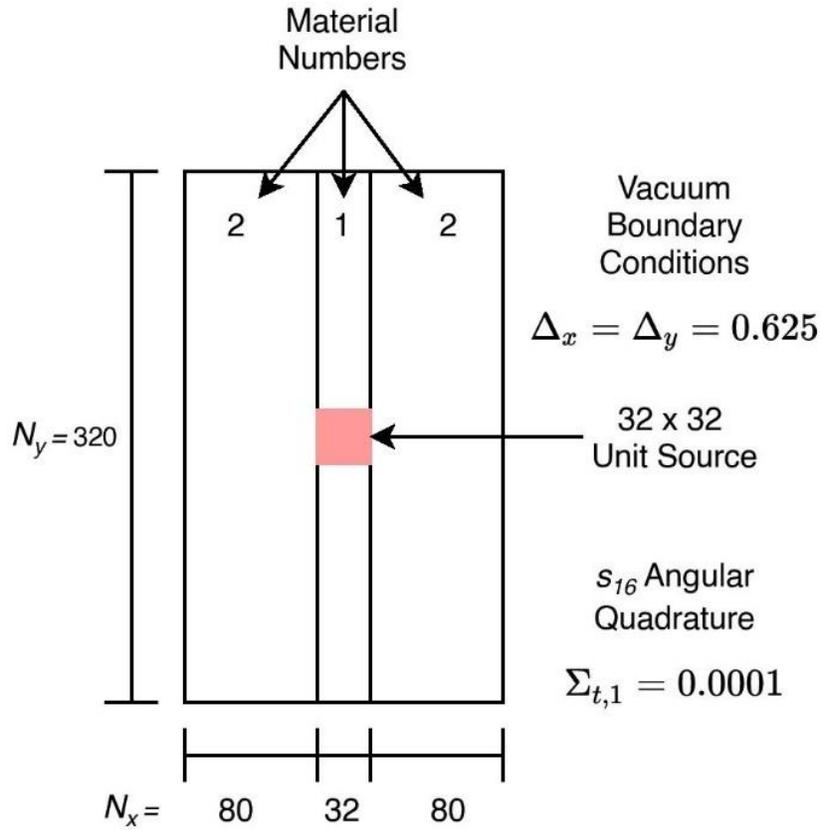


Figure 4.49: Geometry, mesh, and problem parameters for the Straight Duct Problem

Table 4.20: Iterations required to converge Straight Duct Problem

$c =$	0.5			0.9			0.98		
$\Sigma_{t,2} =$	0.1	1.0	5.0	0.1	1.0	5.0	0.1	1.0	5.0
<i>SI</i>	25	54	169	96	270	628	202	829	N/C
<i>DSA</i>	13	30	N/C	24	55	61	30	42	70
<i>AP</i>	13	33	63	24	60	76	N/C	46	96
<i>pNDA</i>	N/C	N/C	N/C	22	N/C	N/C	27	N/C	N/C
<i>PBJ-ITMM</i>	224	199	184	471	335	249	880	629	434
<i>P-PI-SI</i>	22	30	43	82	101	74	172	285	159
<i>AH-PI-SI</i>	131	54	52	356	135	86	704	370	183
<i>IPBJ</i>	232	218	284	535	512	757	N/C	N/C	N/C
<i>P-IP-SI</i>	23	36	93	84	164	330	176	N/C	N/C
<i>AH-IP-SI</i>	145	97	203	428	385	689	858	N/C	N/C
<i>AH-PI-IP</i>	140	81	77	374	163	113	735	400	219

Table 4.21: Runtime observed in seconds to converge Straight Duct Problem

$c =$	0.5			0.9			0.98		
	$\Sigma_{t,2} = 0.1$	1.0	5.0	0.1	1.0	5.0	0.1	1.0	5.0
<i>SI</i>	103	229	706	402	1150	2663	830	3562	N/C
<i>DSA</i>	76	160	N/C	151	314	319	193	247	408
<i>AP</i>	138	267	361	290	576	558	N/C	485	942
<i>pNDA</i>	N/C	N/C	N/C	153	N/C	N/C	190	N/C	N/C
<i>PBJ-ITMM</i>	1448	1310	1239	2992	2277	1653	5512	4269	2880
<i>P-PI-SI</i>	286	390	513	912	1163	841	1834	3128	1768
<i>AH-PI-SI</i>	780	376	348	2049	854	539	4257	2215	1099
<i>IPBJ</i>	534	542	676	1244	1261	1751	N/C	N/C	N/C
<i>P-IP-SI</i>	148	243	612	545	1098	2118	1129	N/C	N/C
<i>AH-IP-SI</i>	338	242	490	1008	964	1625	2001	N/C	N/C
<i>AH-PI-IP</i>	829	534	512	2156	1043	700	4140	2322	1322

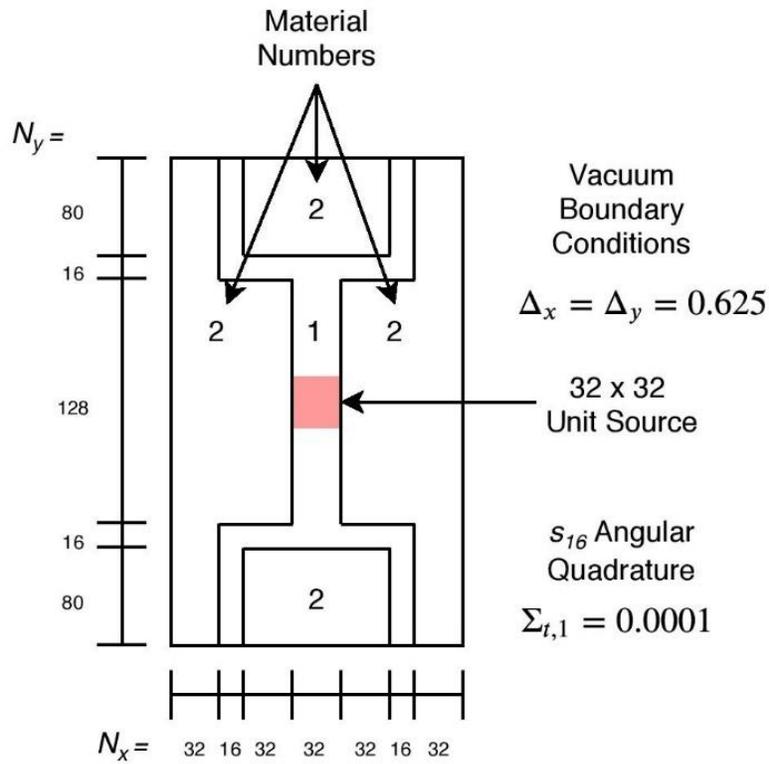


Figure 4.50: Geometry, mesh, and problem parameters for the Dog Leg Duct Problem

Table 4.22: Iterations required to converge Dog Leg Duct Problem

$c =$	0.5			0.9			0.98		
$\Sigma_{t,2} =$	0.1	1.0	5.0	0.1	1.0	5.0	0.1	1.0	5.0
<i>SI</i>	24	54	168	90	263	623	192	801	N/C
<i>DSA</i>	15	28	N/C	28	39	62	37	68	86
<i>AP</i>	15	31	63	28	43	77	37	74	126
<i>pNDA</i>	N/C	N/C	N/C	24	N/C	N/C	27	N/C	N/C
<i>PBJ-ITMM</i>	213	249	242	486	345	377	907	655	456
<i>P-PI-SI</i>	21	29	42	78	98	73	164	276	156
<i>AH-PI-SI</i>	122	55	51	340	131	86	676	360	180
<i>IPBJ</i>	223	260	289	546	487	730	N/C	N/C	N/C
<i>P-IP-SI</i>	21	36	93	79	160	327	167	534	N/C
<i>AH-IP-SI</i>	135	95	201	407	374	682	821	N/C	N/C
<i>AH-PI-IP</i>	131	80	81	358	151	110	707	385	208

Table 4.23: Runtime observed in seconds to converge Dog Leg Duct Problem

$c =$	0.5			0.9			0.98		
$\Sigma_{t,2} =$	0.1	1.0	5.0	0.1	1.0	5.0	0.1	1.0	5.0
<i>SI</i>	105	222	717	383	1120	2579	789	3292	N/C
<i>DSA</i>	99	160	N/C	204	250	330	259	441	548
<i>AP</i>	217	324	432	445	573	781	561	1019	1555
<i>pNDA</i>	N/C	N/C	N/C	190	N/C	N/C	207	N/C	N/C
<i>PBJ-ITMM</i>	1504	1605	1685	3304	2196	2428	5683	4455	3092
<i>P-PI-SI</i>	276	387	524	916	1075	812	1751	2906	1792
<i>AH-PI-SI</i>	714	374	353	2011	753	516	3687	1987	1132
<i>IPBJ</i>	513	633	717	1333	1209	1774	N/C	N/C	N/C
<i>P-IP-SI</i>	135	242	631	535	1027	2101	1142	3106	N/C
<i>AH-IP-SI</i>	315	243	505	1018	927	1592	1986	N/C	N/C
<i>AH-PI-IP</i>	752	519	521	2128	860	640	3857	2121	1254

The configurations of the straight and dog-leg duct problems are shown in Figs. 4.49 and 4.50, respectively. The number of iterations required to satisfy the stopping criterion and the corresponding execution time are listed in Tables 4.20 and 4.21, respectively, for the straight duct problem, and in Tables 4.22 and 4.23, respectively, for the dog-leg duct problem. These results are consistent with the observed trends for the center void problem, even though *PBJ*-type and hybrid methods converge faster for both of these problems across all cases. This is due to the altered problem configuration, with the void region extending to the global boundary, thus increasing the loss of particles via leakage.

## 4.9: Computational Cost Comparison of *ITMM* Matrix Construction Algorithms

In Chapter 3 we discussed the *DMS* algorithm for constructing the *ITMM* matrices and our new alternative, *GFIC*, for its simplicity in extension to unstructured grids. Additionally, we noted that in the ideal circumstance, the inner sweep of either of these algorithms only sweeps over cells that are downstream from the current cell of the outer sweep. Executing this in unstructured grids, however, would require further decomposition of the mesh, thus eliminating the ability to construct the *ITMM* matrices using the existing mesh sweep algorithm of the THOR code. We therefore have two variants of our new *GFIC* algorithm, the ideal sweep that only sweeps over the mathematically required downstream cells, and the full sweep that sweeps over the entire sub-domain, simply assigning zeros to upstream cells by nature of the transport equation with lagged scattering source.

We derived expressions for the scaling of our algorithm's construction time using both sweep types, (3.86) and (3.87), which are proportional to the theoretical construction cost of *GFIC* using the full and ideal sweeps, respectively, under the assumption that the *SI* grind time is independent of sub-domain size. The measured construction times using our HAT-2C code are plotted in Fig. 4.51 along with theoretical trendlines from Eqs. (3.86) and (3.87), using the measured *SI* grind time in HAT-2C.

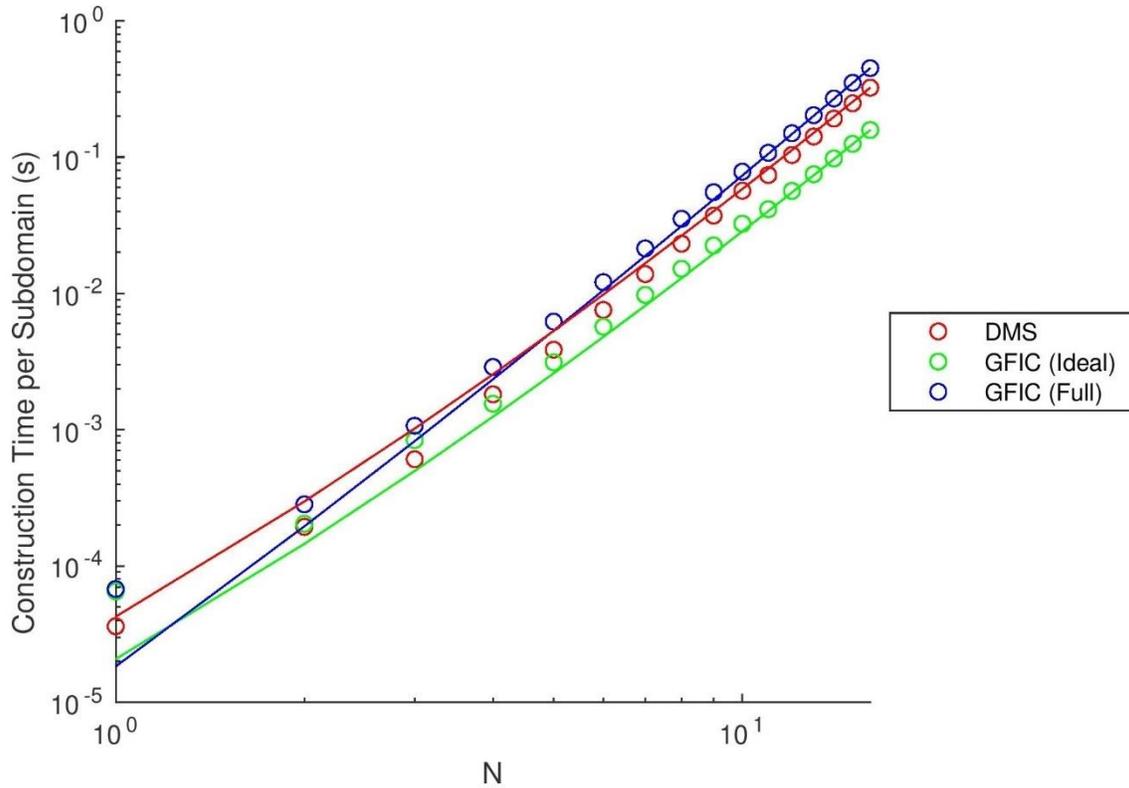


Figure 4.51: Measured (points) and predicted (lines) construction times for *ITMM* matrices using HAT-2C for a single sub-domain using *DMS* and *GFIC* (ideal and full sweep) versus  $N$ , indicating an  $N \times N$  sub-domain size.

The measured construction times for *GFIC* with both the ideal and full sweep follow the predicted trends very closely. Compared to *DMS*, *GFIC* with the ideal sweep is observed to construct the matrices faster. Since these two algorithms sweep over the same number of cells, this indicates that the kernel calculation for *DMS* is more expensive than that of the standard *SI* solution. Additionally, while *GFIC* consumes more time when the full sweep is used compared to when the ideal sweep is used, the measured, per-sub-domain construction times are all below one second, for sub-domains up to  $16 \times 16$  cells in size. Given the advantageous construction times observed and the benefits of implementation using pre-existing code capabilities, *GFIC* with the full sweep was chosen as our *ITMM* construction algorithm to implement into THOR. Note that in order to make our 2-D Cartesian grid studies as indicative as possible of methods' performance in 3-D unstructured grids, *GFIC* with a full sweep was used by HAT-2C for all timing data reported in this chapter.

## 4.10: Summary of Serial Operation Numerical Experiments on 2-D Cartesian Grids

There were seven main objectives for our numerical experiments: (1) verify our HAT-2C code's implementation of the considered iterative schemes; (2) quantify results of our Fourier analyses; (3) conduct homogeneous test problem iterative performance studies; (4) conduct heterogeneous test problem iterative performance studies and motivate the development of the asynchronously hybrid approach; (5) perform a parametric study to elucidate dependencies of the observed iterative performance on problem parameters; (6) test the iterative performance of our hybrid methods on realistic problem configurations containing void regions; and (7) compare *ITMM* matrix construction execution times among the available algorithms.

Iterative capabilities of our HAT-2C code were verified by reproducing convergence rates consistent with previously reported works. Our Fourier analyses of *P-PI-SI* and *P-IP-SI* were then successfully verified. These results showed the error mode at the origin in Fourier space to be the slowest converging for both methods and that the convergence rate is determined by the optically thinner dimension of a problem's cells.

Additionally, while the results of our Fourier analyses demonstrated the robustness of *P-PI-SI* and *P-IP-SI* in optically thin cells, they also suggested that in most homogeneous problems, it would be more advantageous to run only one of the methods comprising the preconditioning method. This was demonstrated by observing the required number of iterations for homogeneous problems. In this study, we found *PBJ* methods in problems with thick cells and *SI* in problems with thin cells to be more advantageous than the corresponding preconditioner method. This prompted a study of the methods' performance in heterogeneous problems and informed the development of the asynchronously hybrid approach, by establishing that executing both methods in most cells is unnecessary.

With all methods developed and a basic understanding of their underlying convergence mechanisms achieved, we conducted a parametric study that assessed the iterative performance of all developed methods and traditional acceleration methods as a function of several problem parameters of a heterogeneous stripe problem configuration. From the results of this study, we concluded that *IPBJ* is less suitable as the primary method in our hybrid approach than *PBJ-ITMM* is. *AH-PI-SI* and *AH-PI-IP* were shown to take significantly longer to converge than *DSA*, *AP* and

*pNDA*, but had runtimes on the same order as these methods, suggesting they are viable approaches for massively parallel solution on unstructured grids, particularly *AH-PI-IP*, as its degree of parallelism matches that of *PBJ-ITMM*. Finally, this study demonstrated our hybrid approach to provide the most acceleration in problems with large regions of very optically thin cells. While *AH-PI-IP*'s increase in iteration counts compared to *AH-PI-SI* became larger as the size of the thin regions increased, these are also the problems for which *AH-PI-SI* has the greatest reduction to the degree of parallelism.

Lastly, we observed the time required by the *DMS* and *GFIC* algorithms to construct the *ITMM* matrices. The construction time for the latter algorithm using a full sweep was shown to be slightly longer than either other option for large sub-domains. However, considering a sub-domain size consistent with what has previously been determined to be the practical size, the construction times between all algorithms are similar, with a modest increase in construction cost associated with *GFIC* when using a full sweep. Thus, *GFIC* with a full sweep is deemed a viable algorithm for *ITMM* matrix construction on unstructured grids due to its ease of implementation in such cases.

# Chapter 5:

## *GFIC* Construction & *PBJ* Solution in Parallel Execution on 3-D Unstructured Tetrahedral Grids

The previous chapter demonstrated the effectiveness of our hybrid approach for accelerating the convergence of *PBJ-ITMM*, a method which is of interest for its application in SDD (Spatial Domain Decomposition) parallel solution on unstructured grids. As discussed in Chapter 3 though, implementation of *PBJ-ITMM* on unstructured grids requires a novel algorithm to construct the associated response matrix operators, with previously used construction algorithms not having simple implementation strategies for unstructured grids. In Sec. 3.2.4, we mathematically formulated an algorithm, *GFIC*, that constructs the *ITMM* matrices independent of the mesh type, using the pre-existing kernel calculation and sweep algorithm typically available in transport codes. The parallelized implementation and testing of *GFIC*, *PBJ-ITMM*, and *IPBJ* on unstructured grids is the topic of this chapter, with *IPBJ* implemented as the currently used alternative to spatially parallel *SI* on unstructured grids, against which we compare the performance of the newly implemented *PBJ-ITMM* on this platform. These methods were implemented in the THOR code [40, 41], a 3-D production-level transport code which solves the short characteristics method with arbitrarily high order local expansion on unstructured tetrahedral grids for both fixed source and  $k$ -eigenvalue problems. The zeroth order local expansion is used in this work. THOR is a steady state code with multigroup and anisotropic scattering capability. Note that because THOR is a multigroup code, in this chapter, we must distinguish between inner and outer iterations. The sections of this chapter are structured as follows.

Our presentation commences with a description of the necessary implementations we have added to the THOR code to enable parallel solution via *PBJ-ITMM* and *IPBJ*. This begins in Sec. 5.1 with a discussion of the process by which we produce a set of local sub-domain THOR meshes from a physical problem geometry. The primary contribution to this process from our work is the developed THOR mesh partitioner, which converts a global THOR mesh to a specified number of sub-domain meshes, each with the information of neighboring sub-domains required for parallel solution. Next, in Sec. 5.2, we discuss the implementation of our novel algorithm for construction of the *ITMM* matrices on unstructured grids, *GFIC*, whose formulation was reported in Sec. 3.2.4.

This discussion focuses on the specific means by which the input parameters are modified by THOR to produce those required by the *GFIC* algorithm. Finally, with the appropriate inputs created and *GFIC* implemented, we discuss the implementation of *PBJ-ITMM* and *IPBJ* in THOR including code flowcharts in Sec. 5.3, with the MPI communication scheme utilized discussed in Sec. 5.4.

Following the presentation of our THOR implementations, we proceed to numerical tests and results. This begins in Sec. 5.5 with a demonstration of the RAM requirements for THOR execution with *PBJ-ITMM*, as memory utilization is of concern due to *PBJ-ITMM*'s matrix solution. Then, the performance of *PBJ-ITMM* and *IPBJ* are tested with a set of scaling tests in Sec. 5.6 on up to 32,768 processors. These tests include weak and strong scaling of the time consumed by each component of the iterative solution process, as well as the scaling of the *GFIC* construction time with respect to sub-domain size. With the parallel performance of *PBJ-ITMM* and *IPBJ* tested, Sec. 5.7 presents the solution CPU time for *PBJ-ITMM* and *IPBJ*, neglecting communication time, compared to the solution time required by the previous serial *SI* version of THOR. This comparison demonstrates the relative total computation requirements for the existing *SI* method and the *PBJ* methods that are proposed as alternatives for spatially parallel solution on unstructured grids.

Next, in Sec. 5.8, we present the performance of *PBJ-ITMM* and *IPBJ* as implemented in THOR on a series of problems containing void regions with varying material properties. These tests demonstrate the iterative slowdown of *PBJ*-type methods in problems containing optically thin cells, justifying the work presented in previous chapters on alleviating this feature with our developed hybrid approach. Finally, the work of this chapter is summarized and conclusions are reached in Sec. 5.9.

## **5.1: THOR Mesh Decomposition Pre-Processing**

Parallel execution of THOR via *PBJ* decomposition requires pre-processing to partition the global mesh into sub-domains. The approach used for this task targeting execution on  $N_p$  processors is to divide the global mesh file into  $N_p$  smaller mesh files with cell faces that lie on processor (sub-domain) boundaries given a separate boundary condition designation. With this approach, each processor simply reads in and solves the cells of its local mesh file, with the

incoming angular fluxes on processor interfaces being treated as boundary conditions that are updated after each inner iteration. In this implementation, we require the number of processors and the number of sub-domains to be equal. Note that in this chapter, adjacent or neighboring processors refer, not to processors that are physically adjacent on the HPC, but to processors which are responsible for the solution of physically adjacent or neighboring sub-domains.

The process by which a set of  $N_p$  sub-domain THOR mesh files are created from a global geometry configuration is depicted by the flow chart in Fig. 5.1.

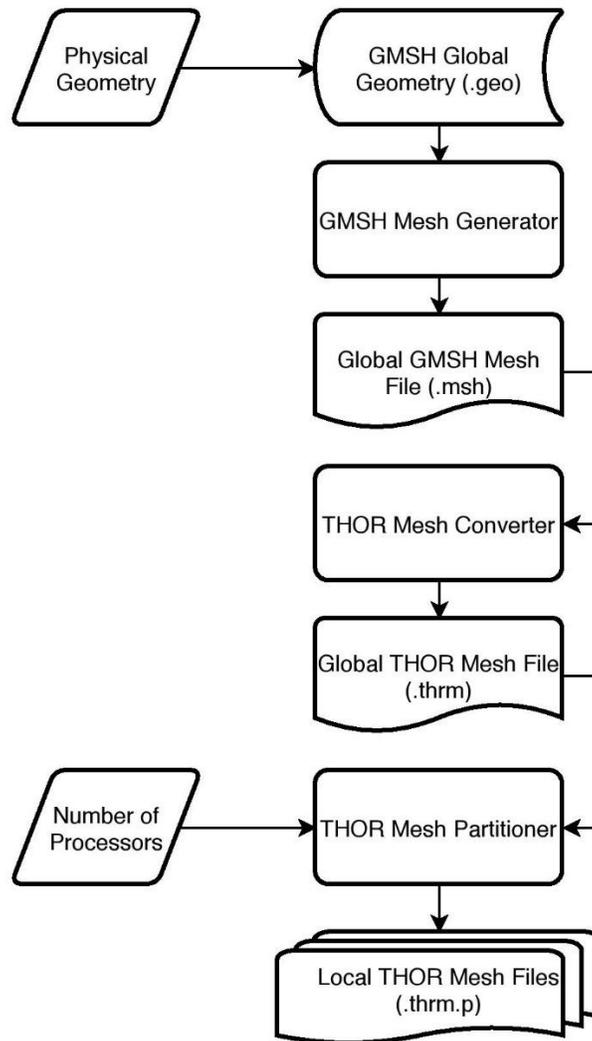


Figure 5.1: Flow chart of process to create partitioned THOR mesh files

The problem’s physical geometry is input to GMSH [68] in the form of a .geo file. GMSH is a mesh generation software tool which we use to create the global mesh. The GMSH mesh

generator is run on the .geo file to create a .msh mesh file of the geometry. The THOR mesh converter, a pre-existing tool in THOR, then converts this mesh to a .thrm file, THOR’s native mesh file format. This global .thrm file is the file that THOR would require for serial execution prior to the implementation of our new solution algorithms. To produce the  $N_p$  local mesh files required for parallel execution with *PBJ* methods, we have created a THOR mesh partitioner, the flowchart for which is in Fig. 5.2. This partitioner produces  $N_p$  .thrm.p files, each containing the local mesh file for the  $p^{\text{th}}$  sub-domain.

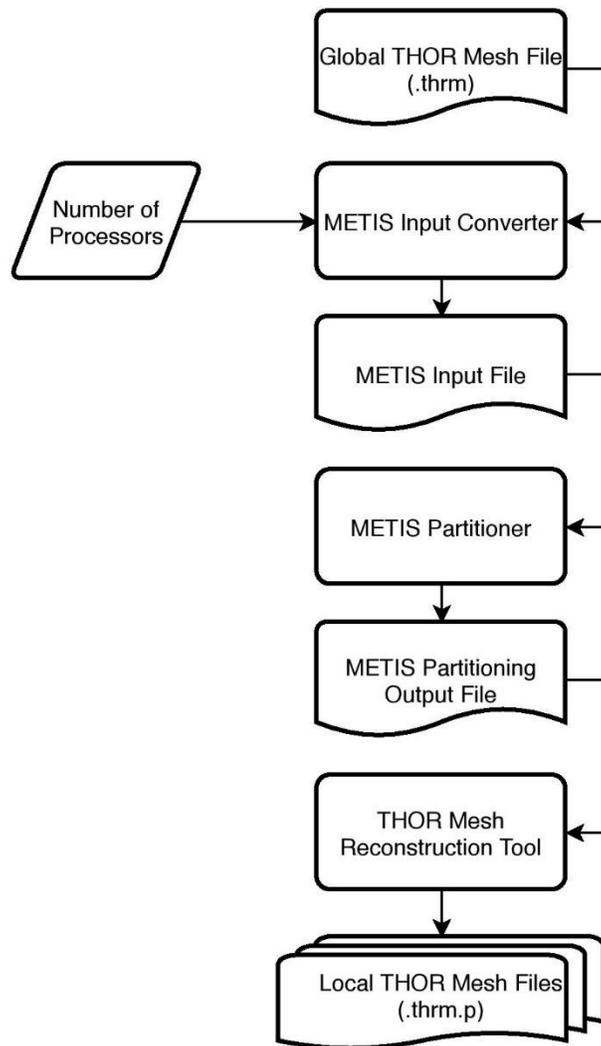


Figure 5.2: Flow chart of THOR mesh partitioner tool

Our THOR mesh partitioner tool converts the global THOR mesh into  $N_p$  local sub-domain THOR meshes. The partitioning is performed using METIS [69], an open-source software tool for

partitioning graphs and meshes. Our mesh partitioner converts the contents of the global mesh .thrm file and the desired number of processors to a METIS input file, then executes the METIS partitioner on this input file, producing the METIS partitioning output file. Our THOR mesh reconstruction tool converts this METIS output, which simply lists cell numbers with their associated processor numbers, to  $N_p$  .thrm.p files. These local mesh files are of similar format to the global mesh file used for serial execution, but with a few pieces of additional data. Firstly, a THOR mesh file contains the number of cells and the number of vertices (points, four of which make up one tetrahedron). The local .thrm.p files contain these values both for the local sub-domain as well as the global problem. Additionally, there is another valid boundary condition specifier for faces, a sub-domain boundary interface. Finally, the adjacency list contains one extra entry per line. The adjacency list typically consists of lines with four entries representing two cell and face number combinations that are the same face, indicating neighboring cells. The extra per-line entry in .thrm.p files indicates the processor to which a neighboring cell belongs. These pieces of additional information allow a processor to execute all required tasks, reading only its local mesh file.

For all partitioned meshes utilized in the chapter, the tolerance to partitioning load balance was set to 3%, instructing METIS to produce a partitioning in which the largest and smallest sub-domains are within  $\pm 3\%$  of the average sub-domain size, measured in number of cells.

## 5.2: *GFIC* Implementation in THOR

We have updated THOR with capabilities to solve problems via either *PBJ-ITMM* or *IPBJ* using the previously discussed  $N_p$  sub-domain mesh input files. To solve using *PBJ-ITMM*, we implemented the *GFIC* algorithm formulated in Sec. 3.2.4 into THOR. During the pre-solve phase, *GFIC* is the primary process executed when *PBJ-ITMM* is the solution method used. However, instructions for the processors' MPI communications are also generated and transmitted to adjacent processors during this phase, whether *PBJ-ITMM* or *IPBJ* is the specified solution method. For a full description and mathematical formulation of *GFIC*, refer to Sec. 3.2.4.

To implement *PBJ-ITMM* in THOR, we effectively execute the program in its entirety twice, under different conditions. We refer to these as the pre-solve and solve phases. During the pre-solve phase, each processor reads in its mesh, recalling that this mesh is the portion of the

problem which is directly accessible to that individual processor, a single sub-domain. Note that this process occurs in parallel over all processors, each for its associated sub-domain. We store the actual sub-domain boundary conditions, external source, and maximum number of iterations (both outer and inner) to temporary variables and impose values to all of these variables to transform the transport solver into a *GFIC* routine. First, to create the “zero” state, we set all boundary conditions to vacuum and the external source to zero. Note: THOR automatically uses a zero initial guess. Were the code to accept a non-zero initial guess, this process would be required for the initial guess as well. Logical tests are used to skip all calculations which add contributions to the external source, such as fission, between-group scattering, and between angle scattering. Additionally, for optimized performance, all unnecessary tasks, such as convergence checks are skipped during the pre-solve phase.

In the *GFIC* stage, the maximum number of outer iterations is set to one and the maximum number of inner iterations is set to  $N_{cells} + N_{BF}$ , where  $N_{cells}$  is the number of cells in a processor’s domain and  $N_{BF}$  is the total number of boundary faces in the processor’s sub-domain.  $N_{cells}$  is the number of inner iterations required to construct  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$  and  $N_{BF}$  is the number of inner iterations required to construct  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$  (since each inner iteration loops over all angles). With the single outer iteration looping over energy groups, this is therefore the correct iteration structure to construct the *ITMM* matrices for all groups.

With the “zero” state initialized and the correct iteration structure setup, the bulk of *GFIC* is implemented by using logical statements to impose unit fluxes to the correct cell or face and to store the resulting data appropriately in the target *ITMM* matrix. For the first  $N_{cells}$  inner iterations, a unit cell-averaged scalar flux is imposed to a different cell each iteration for  $\mathbf{J}_\phi$  and  $\mathbf{J}_\psi$  construction. Afterwards, a unit incoming angular flux is imposed on a single sub-domain boundary face for  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$  construction, with the discrete ordinate of the imposed flux changing after each angle is swept, rather than only after each inner iteration. Note that since, typically, half of the angles will be outgoing for a given face, the mesh sweep is skipped for angles that are outgoing or parallel to the given face on which unit fluxes are being imposed. After each iteration (or individual angle sweep for  $\mathbf{K}_\phi$  and  $\mathbf{K}_\psi$ ), the outgoing angular fluxes on the processor domain’s boundary are stored to a column of  $\mathbf{J}_\psi$  (or  $\mathbf{K}_\psi$ ) and the cell-averaged scalar fluxes are stored to a column of  $\mathbf{J}_\phi$  (or  $\mathbf{K}_\phi$ ).

In Cartesian grids, ordering the fluxes within these matrices is trivial due to the structured nature of the sub-domains. In unstructured grids, however, there is no natural ordering to cells and edges. For our implementation, we index the *ITMM* matrices based on the arbitrary ordering of the cells and faces utilized by THOR. For the sub-domain boundary angular fluxes, this ordering is saved to vectors for mapping. These vectors are referred to as instructions, as they inform which value an entry in an angular flux vector represents. Consequently, these vectors can be used to instruct which processor entries from  $\boldsymbol{\psi}_{out}$  are to be sent to and the locations in  $\boldsymbol{\psi}_{in}$  to which values received from processors are to be assigned. As this communication is required for both *PBJ-ITMM* and *IPBJ*, these instructions must be generated regardless of which method is specified. Therefore, when *IPBJ* is used, the pre-solve phase still occurs, but without performing the *ITMM* matrix construction operations. The result is simply a set of sweeps without any computations, generating the needed communication instructions.

These communication instructions are used to create packing instructions for a processor, designed to separate the  $\boldsymbol{\psi}_{out}$  vector into  $N_{neigh}$  smaller vectors,  $N_{neigh}$  being the number of sub-domains that border a given processor's sub-domain. The relevant portions of the communication instructions are then exchanged between adjacent sub-domains using MPI. The received communication instructions are used by the receiving processor to create unpack instructions. These unpack instructions map the values from the vectors that will be received each iteration from adjacent processors to populate  $\boldsymbol{\psi}_{in}$  during the *PBJ*-type iterations.

Another consequence of the non-trivial indexing of the *ITMM* matrices and vectors is the sparse storage of  $\mathbf{K}_{\boldsymbol{\psi}}$ . Recall that  $\mathbf{K}_{\boldsymbol{\psi}}$  is frequently the largest of the *ITMM* matrices, however, it is very sparsely populated. In unstructured grids it is not trivial to pre-calculate the number of non-zero entries in  $\mathbf{K}_{\boldsymbol{\psi}}$ , as the irregularly shaped sub-domains make determination of the number of boundary outgoing angular fluxes that are downstream from a given boundary incoming angular flux difficult. Additionally, the location of these non-zero elements within  $\mathbf{K}_{\boldsymbol{\psi}}$  is not easily predeterminable. The most obvious technique for overcoming this obstacle is to construct the full  $\mathbf{K}_{\boldsymbol{\psi}}$  matrix, count the number of non-zero values, then transfer these values along with their respective locations within the matrix to sparse storage. While this prevents unnecessary calculations when computing  $\mathbf{K}_{\boldsymbol{\psi}}\boldsymbol{\psi}_{in}$  each iteration, when we initially implemented this strategy, it was found that the memory consumption by the full  $\mathbf{K}_{\boldsymbol{\psi}}$  matrix even in this temporary capacity

was excessive, severely limiting the scope of problems to which *PBJ-ITMM* was applicable. To mitigate this memory consumption issue, we implemented a dynamic strategy that constructs  $\mathbf{K}_\psi$  in sparse storage directly. This strategy begins with a guessed number of non-zero elements, determined by the number of cells in the sub-domain and the number of angles in the quadrature. When a  $\mathbf{K}_\psi$  element is to be stored, if the sparse storage is full, it is reallocated to double its current size. At the conclusion of the construction process, the completed sparse  $\mathbf{K}_\psi$  matrix is then reallocated to its final size, eliminating any trailing zeros. This process mitigates excessive memory consumption when constructing  $\mathbf{K}_\psi$  without having to reallocate it in memory an exorbitant number of times, which can be costly. Note that for multigroup problems, this process only occurs when constructing the *ITMM* matrices for the first group, as the number and location of non-zero values does not change between groups for a given angular quadrature.

Upon completion of the *GFIC* algorithm, the *LU* factorization of  $(\mathbf{I} - \mathbf{J}_\phi)$  is computed and stored. Afterwards, all *ITMM* matrices, the *LU* factorization of  $(\mathbf{I} - \mathbf{J}_\phi)$ , which doesn't change between iterations in the same energy group, and instructions for packing and unpacking angular flux vectors before and after each communication step, are constructed for all energy groups. The pre-solve phase then terminates by reverting all artificially imposed parameters, namely, maximum number of outer and inner iterations, boundary conditions, and external source, to their original values. The pre-solve phase then ends, returning to the beginning of the THOR code to commence the solve phase.

### 5.3: *PBJ-ITMM* and *IPBJ* Implementation in THOR

The solve phase utilizes the same iteration structure already present in THOR. In this structure, each outer iteration loops over energy groups, calculating the production in the current energy group from other groups to create a between-group external source. The inner iterations then run to converge the within-group problem. For solution via *PBJ-ITMM*, the inner iterative solution is modified from a series of mesh sweeps to the *ITMM* matrix operations and, for parallel execution, this iterative solution is followed by packing operations and communication. Additionally, communication is required for global calculations such as the convergence check

and the calculation of  $k_{eff}$ , which are performed using the MPI\_ALLREDUCE function. The inner iteration process for *PBJ-ITMM* is depicted in Fig. 5.3.

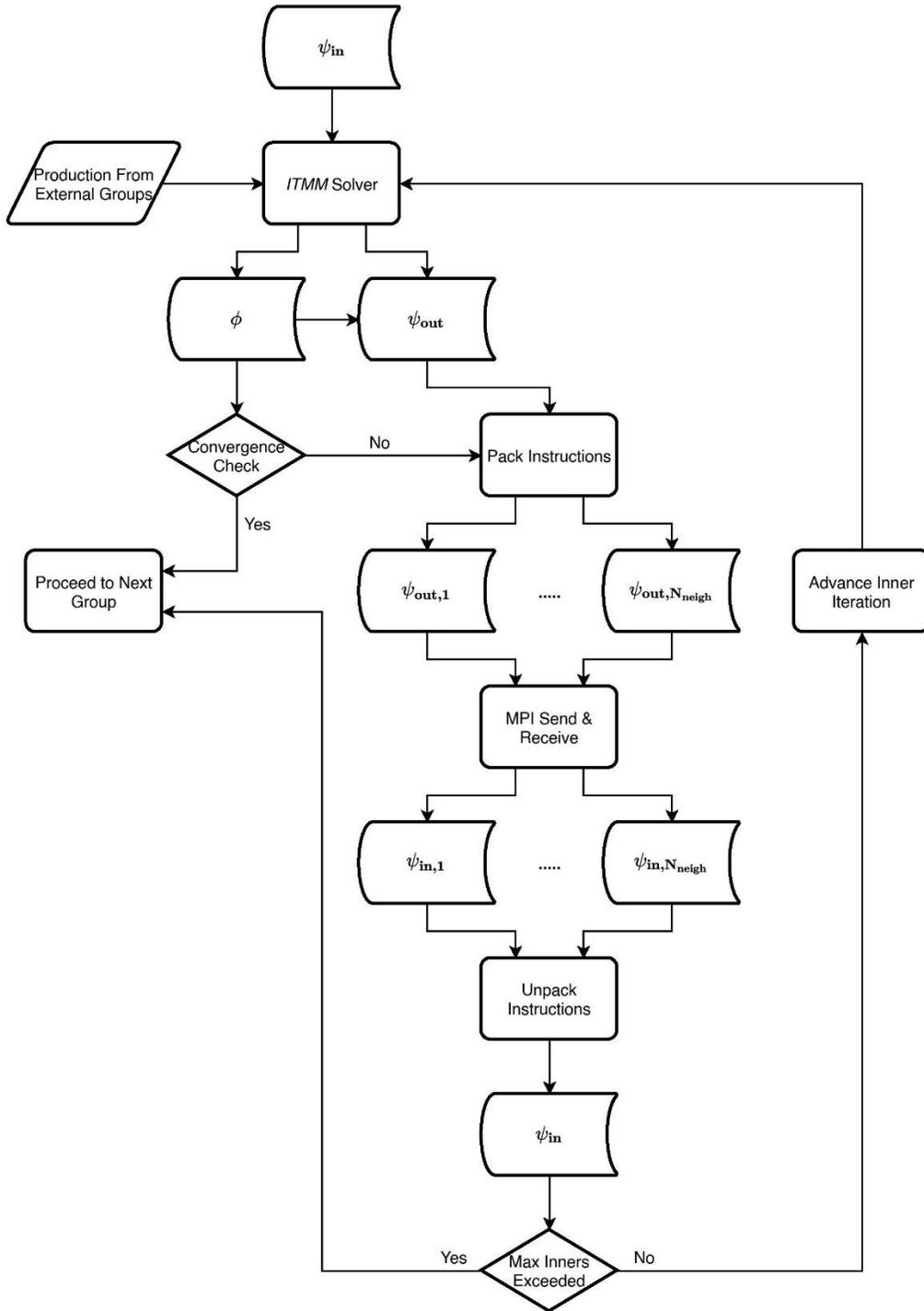


Figure 5.3: Flow chart of *PBJ-ITMM* inner iteration process on a single processor in THOR

This is a typical parallel inner iterative process as executed on a single processor in parallel with the same on all participating processors, with a solver followed by a convergence check and communication, repeated until the stopping criterion is satisfied or the maximum number of inner iterations is exceeded. Note that exceeding the maximum number of iterations does not mean that the program terminates unsuccessfully. Rather, it progresses to the next group. Since the inner iteration process comprises the bulk of the required computational load, it is common practice to set the maximum number of inner iterations low, thus allowing the production from external groups to be updated more frequently, reducing the total number of inner iterations required for convergence. The pack and unpack operations are performed using the instructions created during the pre-solve phase, and the MPI send and receive operation is discussed in Sec. 5.4.

The *ITMM* solver in THOR is a newly developed solution module which replaces the traditional mesh sweep with the *PBJ-ITMM* method for obtaining the sub-domain iterative solution. This module solves Eq. (3.56) to obtain the vector of cell-averaged scalar fluxes, followed by Eq. (3.57) to obtain the vector of sub-domain boundary outgoing angular fluxes. The external source in these equations includes the production from all other groups, a term which updates after each outer iteration. The incoming angular flux vector is obtained by global boundary conditions, the initial guess, or communication from adjacent processors. The solution of Eq. (3.56) utilizes the *LU* factorization of  $(\mathbf{I} - \mathbf{J}_\phi)$  computed during the pre-solve phase, solving the matrix equation using LaPACK. [63]

*IPBJ*'s implementation in THOR follows a similar inner iterative structure as *PBJ-ITMM*, depicted for a single processor in Fig. 5.4.

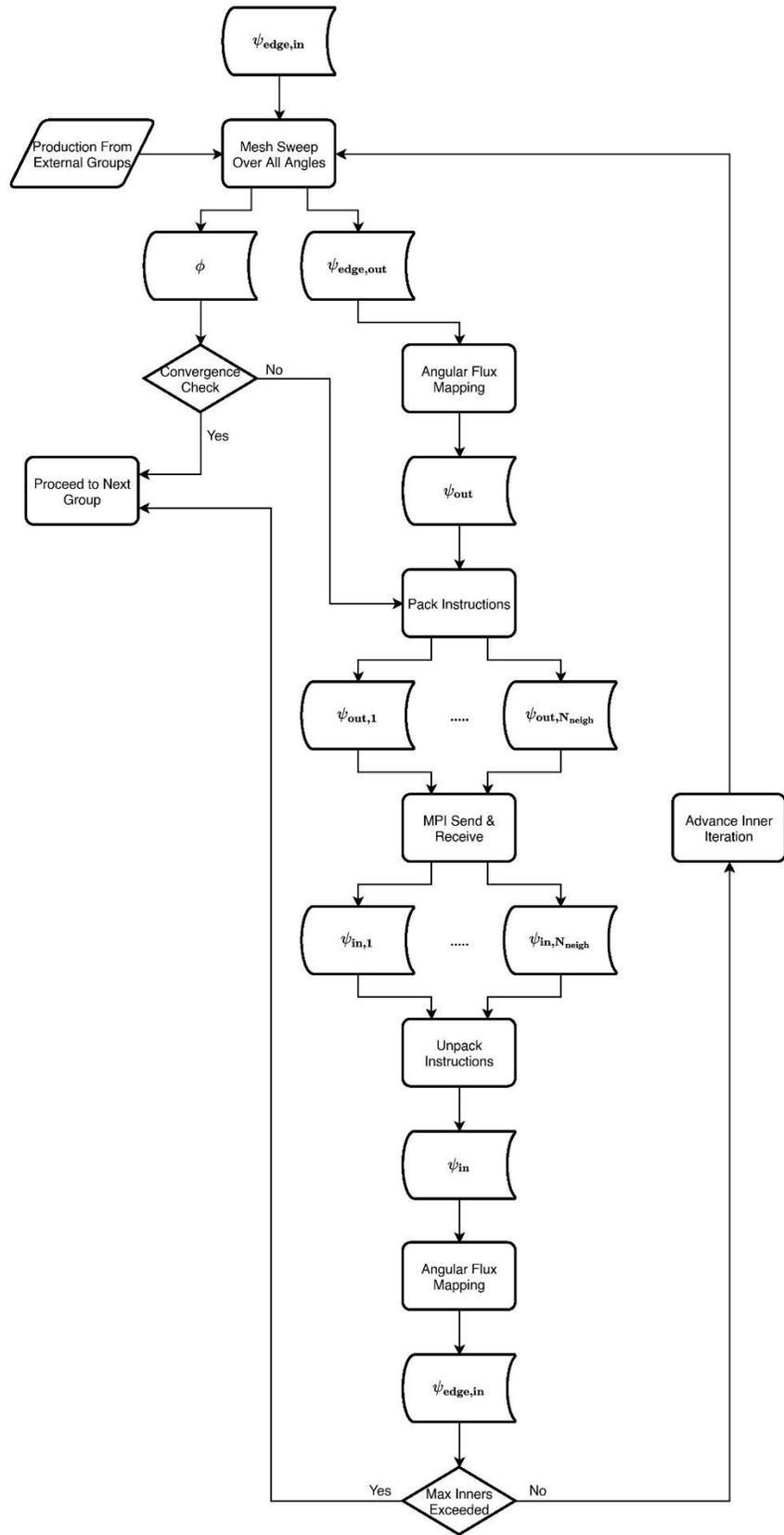


Figure 5.4: Flow chart of *IPBJ* inner iteration per process in THOR

The inner iterative process for *IPBJ* in THOR is similar to that of *PBJ-ITMM*, except for two key differences. Firstly, the *ITMM* solver is replaced by calling the pre-existing mesh sweep routine, executing a sequence of mesh sweeps over the processor’s sub-domain. Additionally, in the communication phase, there are two “angular flux mapping” stages. Since *IPBJ* is not a matrix solution, it produces cell-edge angular fluxes for all cells, rather than the compact vector of sub-domain boundary outgoing angular fluxes produced by *PBJ-ITMM*,  $\psi_{out}$ . The outgoing angular fluxes of cell faces that are on the sub-domain boundary,  $\psi_{edge,out}$  are mapped to the proper locations in  $\psi_{out}$  based on the order of indices that was assigned in the pre-solve phase. Likewise, *IPBJ* does not use the input of a  $\psi_{in}$  vector, but rather, requires the incoming angular fluxes on cell edges that lie on the sub-domain boundary to be known. Therefore, the  $\psi_{in}$  vector that is unpacked after the communication phase is mapped to  $\psi_{edge,in}$ , the cell-edge angular fluxes that lie on the sub-domain boundary and are incoming to the sub-domain, based on the order prescribed during the pre-solve phase.

These *PBJ-ITMM* and *IPBJ* implementations utilize pre-existing capabilities of THOR; capabilities which are also present in most unstructured grids  $S_N$  transport codes. While *PBJ-ITMM* replaces the solver routine, the implemented solver is a simple matrix solution. Because the matrices required for this solution are constructed with *GFIC* using the pre-existing kernel calculation and solution algorithm, the construction is agnostic to the codes’ other features, such as the spatial discretization method. Consequently, the implementation of this entirely different iterative method does not require re-coding functionality that developers of production codes may have invested substantial amounts of programming effort and time into developing and verifying. Rather, due to *GFIC*’s utilization of the pre-existing mesh sweep, the *ITMM* matrices are constructed to model the same physics that the standard *SI* solution would. This feature greatly increases *GFIC*’s desirability for implementation in production codes, as it does not require large overhaul of the existing code.

## 5.4: MPI Implementation for Processor Communication

The inter-processor communication required for both *PBJ-ITMM* and *IPBJ* is performed using Message Passing Interface (MPI) library routines. MPI executes  $N_p$  (number of processes) instances of a program, assigning each instance a different MPI rank, and allowing messages to be

sent and received based on this rank identifier. In this work we avoid overloading, so the number of processors on which THOR runs in parallel is also  $N_p$ . Codes using MPI for parallelism must therefore be designed to specify the tasks assigned to a given processor based on this rank. For THOR, this rank indicates the partitioned mesh that will be read by a processor, assigning each processor a sub-domain. MPI communication is then used for the primary function of communicating sub-domain interface angular fluxes between processors after each inner iteration, packaged into one message per neighbor. The location of this communication within the inner iterative algorithms for *PBJ-ITMM* and *IPBJ* are shown in Figs. 5.3 and 5.4 respectively. With  $N_{neigh}$  angular flux vectors packaged for transmission, the communication between adjacent processors is accomplished using `MPI_ISEND` and `MPI_IRECV`. These functions send and receive, respectively, data to or from a processor of specified rank that corresponds to a neighboring sub-domain. Consequently, using these functions for all neighboring processors achieves the communication required for either *PBJ-ITMM* or *IPBJ*, noting that the communication requirements are identical for the two methods. Also, note that in unstructured grids with sub-domains comprising more than one cell per sub-domain, the number of neighbors is not fixed, hence the communication penalty is not uniform across processors.

The “I” preceding “SEND” or “RECV” specifies non-blocking communication. Blocking communication dictates that the code will not progress past a send command until the processor is notified that its sent message has been received. Likewise, the code will not progress past an receive command until the requested message arrives at its destination. Conversely, non-blocking communication specifies that `MPI_ISEND` will send the message, which MPI stores in buffer memory until it is requested by the target processor, and the sending processor will proceed to execute through the code without waiting for the message exchange to complete. Likewise, `MPI_IRECV` submits a request to MPI for a message from a processor. The requesting process will continue executing code though, regardless of whether or not the message has arrived.

Non-blocking communication is used because of the arbitrary ordering of neighboring processors associated with unstructured grids. With nothing to guarantee that neighboring processors will attempt to communicate with each other at the same time, deadlock cycles become inevitable if blocking communication is used. To illustrate this concept, consider three processors (P1, P2, and P3), all of which neighbor the other two. If P1 first attempts to communicate with P2, P2 with P3, and P3 with P1, then blocking communication will deadlock, with all three processors

waiting for communication with a processor which will never resolve. Non-blocking communication is therefore used to avoid this complication. Specifically, a processor first calls `MPI_IRecv` for all adjacent processors, requesting messages. The receive function is called first so that messages are received and removed from the buffer as quickly as possible after they are sent. Processors then call `MPI_Isend` for all adjacent processors. Afterwards, `MPI_Waitall` is called. This function forces a processor to wait until all sent messages are received and all receive requests are filled. This is performed on all participating processors so that the following iteration does not begin without updated incoming angular flux information.

While transfer of sub-domain boundary angular fluxes between iterations is the primary MPI communication, there are a couple of other MPI communications used in THOR as well. Generally, these communications are used for global calculations and processes. Specifically, in  $k$ -eigenvalue problems, `MPI_Allreduce` is used after each outer iteration to update the multiplication factor (criticality). `MPI_Allreduce` sends a set of data to the root processor, which performs a specified operation on the data (summation, maximum/minimum value, etc.), the result of which is sent back to all processors. For calculation of  $k_{eff}$  specifically, the sub-domain fission production is sent with the `MPI_Allreduce` function using the SUM operation, returning the global fission production as required for calculation of criticality. The `MPI_Allreduce` function is also used after each iteration (inner or outer) to determine convergence, sending each sub-domain's maximum relative iterative change in a tested quantity with the MAX function to determine the maximum relative iterative change in this quantity across all processors. Convergence must be assessed globally because otherwise, if all sub-domains do not converge in the same iteration, which they usually do not, then processors on which convergence has been observed will terminate the current iterative sequence, while other processors continue to iterate and become deadlocked, waiting for messages from processors that have converged and are no longer iterating. Finally, MPI functions are used after the solver completes its work for postprocessing output quantities such as calculating region-wise scalar fluxes.

## 5.5: *PBJ-ITMM* Memory Requirements

When implementing *PBJ-ITMM*, memory (RAM) requirements associated with storing the response matrices is of concern, [55] as the sizes of these matrices scale super-linearly with the number of cells per sub-domain and bilinearly with the number of cells per sub-domain and quadrature angles. In our implementation of *PBJ-ITMM* in THOR, each processor is required to store all four *ITMM* matrices associated with its sub-domain for each energy group, with  $K_\psi$  utilizing the sparse construction and storage strategy discussed in Sec. 5.2. To quantify the total memory requirement associated with *PBJ-ITMM* in THOR, we execute a series of problems, with the Linux operating system reporting the maximum amount of RAM allocated by the program at any time during a run. This quantification therefore represents the total amount of memory per processor required to execute THOR using *PBJ-ITMM*, not just to construct and store the *ITMM* matrices.

The problem used for this test is the Godiva benchmark. A full description of this problem configuration is available in Sec. 5.6.1. However, for testing the memory requirement, the problem configuration is largely irrelevant, as the required amount of RAM is dependent on the number of cells per sub-domain, the number of energy groups, and the quadrature order, not the material composition of cells. The maximum memory allocated per processor by THOR when using *PBJ-ITMM* to solve Godiva for a variety of sub-domain sizes and quadrature orders is presented in Table 5.1, with sub-domain's containing cell counts of approximate powers of two. While the number of energy groups is not varied, all data structures of significant size scale linearly with respect to the number of groups. Godiva is a six-group problem. To ensure that memory usage associated with MPI communication is accounted for, all cases are executed with 16 processors. While the number of cells in a sub-domain varies slightly across processors, the memory requirement is collected from the root processor. We therefore provide the number of cells in the sub-domain allocated to the root processor. Note that any value with one \* indicates the case was executed only on eight processors and any value with two \*\* indicates the case was executed only on four processors. These processor counts are reduced for selected cases due to the limited memory available on the desktop computer on which this test was conducted.

Table 5.1: RAM (GB) per processor consumed by THOR to solve Godiva using *PBJ-ITMM* with  $N$  cells per sub-domain with varying quadrature orders

$N$	$S_4$	$S_6$	$S_8$	$S_{12}$	$S_{16}$
8	0.014	0.014	0.015	0.016	0.018
16	0.015	0.016	0.017	0.021	0.027
31	0.016	0.018	0.021	0.029	0.040
62	0.021	0.027	0.036	0.061	0.095
131	0.041	0.067	0.103	0.195	0.323
244	0.090	0.158	0.251	0.502	0.846
500	0.257	0.470	0.757	1.533	2.600
988 / 984* / 988**	0.786	1.986	2.332	4.872*	8.240**

Table 5.1 provides measurements that can be used to estimate the amount of RAM one would expect to require to solve a problem with *PBJ-ITMM* given the quadrature order and number of cells per sub-domain. The primary conclusion from this data is that on modern HPC and desktop platforms memory requirements are, in general, not a severe barrier for *PBJ-ITMM*. However, there are certain conditions under which the required memory can become problematic. The supercomputer used for the scaling tests in subsequent sections is Sawtooth [70] at Idaho National Laboratory. A single node of Sawtooth is comprised of 48 Intel Xeon® Platinum 8268 cores and 196 GB of RAM. Sawtooth’s network is a 9-D enhanced hypercube using EDR and HDR InfiniBand. On such a platform, only the cases with  $\sim 1024$  cells per sub-domain and  $S_{12}$  or  $S_{16}$  quadrature would be expected to exceed the available memory. Our testing of THOR on Sawtooth utilizes an  $S_4$  quadrature, the associated memory requirements for which are well below the available RAM.

For applications extending past our ensuing tests, though, the memory requirement must be taken into account. For higher order quadratures,  $\sim 1024$  cells per sub-domain would likely exceed the memory available on an HPC system. Recall, however, that the optimal number of cells per sub-domain for *PBJ-ITMM* was previously estimated to be 64. [55] While the optimal number will inevitably change with HPC architecture and has likely changed since the time of publication of [55] due to hardware improvements, 1024 cells per sub-domain is still expected to be well above the optimal size. Hence, it is used as the maximum sub-domain size for our tests because it was far enough above previous estimates of the optimal size that it was assumed it would exceed the

range of sub-domain sizes over which *PBJ-ITMM* is effective. Therefore, while 1024 cells per sub-domain is included in our testing suite, it is a sub-domain size that is not likely to be used for application, and the associated increase in the required memory is not a significant shortcoming.

One foreseeable manner in which the required memory can become of concern though, is in problems with a large number of energy groups. For modest quadratures such as  $S_6$ , memory would still likely not be of concern, as the number of groups would have to greatly exceed 300 before the Sawtooth system's available memory would be expended, assuming linear scaling of the required memory with number of groups and  $\sim 128$  cells per sub-domain. At increased quadrature orders, however, problems with 100+ energy groups will likely require more RAM than is available, even at sub-domain sizes of  $\sim 128$  cells. There are mitigation strategies for these scenarios though. One strategy is to implement *PBJ-ITMM* such that each processor solves multiple sub-domains analogous to the Red/Black Gauss-Seidel iterative scheme reported in [52]. Due to the nonlinear scaling of the *ITMM* matrices' sizes, for a given mesh size and processor count, this would reduce the memory requirement. Additionally, when solving the inner iterations for a group, only the *ITMM* matrices associated with that group are needed. Therefore, it is possible to write the *ITMM* matrices to files, having the program store in memory only the matrices for a certain number of groups simultaneously. To avoid excessive strain on the file system that would be caused by all processors opening and reading files repeatedly, a staggering approach can be used, having different processors read in new matrices between different energy groups. While this approach would be costly due to the repeated file reading, this obstacle is only foreseen in problems with very high angular and energy resolution, using a method that is intended for use with multi-million cell problems. These are therefore extremely computationally intensive problems that will benefit greatly from massively parallelized calculation.

The main finding of these memory requirement tests is that RAM utilization is not expected to be of concern for our testing suite, nor is it expected to be a major issue for most applications. Additionally, for the specific applications mentioned for which RAM utilization may be a significant issue, mitigation strategies exist.

## 5.6: *PBJ-ITMM* and *IPBJ* Scaling Tests with THOR

To test the performance of our *PBJ-ITMM* and *IPBJ* implementations on unstructured tetrahedral meshes using parallel processing, we perform strong and weak scaling, and evaluate the scaling of *GFIC*'s construction time, on two test problems, Godiva [71] and C5G7 [72]. Strong and weak scaling are two traditional approaches for assessing the parallel performance of a code. Strong scaling studies the trend of a program's execution time for a constant amount of work with increasing processor count. The objective of strong scaling is thus to determine the degree to which a program can parallelize a problem and the minimum amount to which the per-processor work can be reduced and still maintain parallel efficiency. By continually adding more processors to the same problem, the amount of work per processor decreases until it reaches a level that cannot be further reduced. The time associated with performing this work is referred to as the serial cost of a program, as it represents the portion of the total work that cannot be conducted in parallel. By identifying this serial cost, strong scaling provides a quantification of how much the execution time of a program can be diminished for a given problem with an excessive amount of computational resources allocated.

Contrarily, with weak scaling as the number of processors is increased, the total amount of work is increased proportionately, thus keeping the amount of work per processor constant. The objective of weak scaling is thus not to determine the degree to which a code can parallelize the solution process for a fixed problem, but to observe the performance of a program as problem size increases. With the amount of work per processor constant, the ideal weak scaling trend is a horizontal line, with execution time constant as the problem size and processor count increase proportionality to each other. However, as processor count increases, sources of parallel inefficiency inevitably arise, such as increased communication costs, rendering weak scaling trends of increasing execution time as processor count increases. The rate at which this execution time increases dictates the size of problem that a code is effective at solving. The combination of these two scaling tests provides a broad view of a program's parallel effectiveness.

A specific detail of strong and weak scaling that must be addressed for our studies is the exact definition of "work." Work can be defined in multiple ways, each of which results in a different scaling test. For our studies, our interest is in the iterative properties of *PBJ-ITMM* and *IPBJ* as well as their scalability. Since quadrature order and number of groups will be constant

across a scaling trend, we define work as being proportional to the number of cells. Consequently, the effects of *PBJ-ITMM* and *IPBJ*'s iterative properties, mainly their iterative slowdown as sub-domains become thin, are not neglected from our results.

All problems solved in this section are eigenvalue problems that utilized an  $S_4$  angular quadrature, an inner iteration relative stopping criterion of  $10^{-7}$ , and outer and  $k$ -eigenvalue relative stopping criteria of  $10^{-6}$ . Additionally, for all scaling tests, we use a maximum number of inner iterations equal to two. This means that in an outer iteration, only a maximum of two inner iterations per group will be executed. This is a common strategy for executing the nested inner-outer iterations in a multi-group calculation with up-scattering as it reduces the total number of inner iterations consumed, with the between-group production updated more frequently than if the inner iterations were executed to convergence within each outer iteration.

### 5.6.1: Godiva Scaling Results

Our first test problem used for strong and weak scaling is Godiva [71], a critical homogeneous sphere of highly enriched Uranium (HEU), the cross-sections for which have six energy groups and isotropic scattering. For scaling tests on Godiva, we impose upper and lower limits on the mesh size, number of processors, and number of cells per sub-domain. Then our testing suite is produced by every combination of each power of two that falls between the bounds of each parameter's range. For number of cells per sub-domain, we impose a minimum value of 8 and a maximum value of 1024 ( $2^{10}$ ). The minimum value is chosen because there is a lower limit on the sub-domain size that METIS can reliably partition into. The maximum value is chosen because it significantly exceeds what we expect to be the ideal sub-domain size for *PBJ-ITMM*. The minimum and maximum number of processors are 1 and 32,768 ( $2^{15}$ ), respectively. This upper limit is based on the availability of HPC resources, not on observed limitation of either method. Finally, the minimum and maximum mesh sizes are 8 and 8,388,608 ( $2^{23}$ ), respectively. The lower mesh size limit is based on the minimum number of cells per sub-domain as well as the minimum mesh size obtainable from GMSH. The maximum mesh size is based on the maximum number of processors and 256 cells per sub-domain. Larger meshes were not constructed due to the limited number of data points that could be tested on them given our processor limit of 32,768. These limits result in the testing suite for Godiva presented in Table 5.2.

Table 5.2: Godiva scaling test suite

Target Mesh Size	Number of Processors (Sub-Domains)							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	1
16 ( $2^4$ )	-	-	-	-	-	-	1	2
32 ( $2^5$ )	-	-	-	-	-	1	2	4
64 ( $2^6$ )	-	-	-	-	1	2	4	8
128 ( $2^7$ )	-	-	-	1	2	4	8	16
256 ( $2^8$ )	-	-	1	2	4	8	16	32
512 ( $2^9$ )	-	1	2	4	8	16	32	64
1024 ( $2^{10}$ )	1	2	4	8	16	32	64	128
2048 ( $2^{11}$ )	2	4	8	16	32	64	128	256
4096 ( $2^{12}$ )	4	8	16	32	64	128	256	512
8192 ( $2^{13}$ )	8	16	32	64	128	256	512	1024
16,384 ( $2^{14}$ )	16	32	64	128	256	512	1024	2048
32,768 ( $2^{15}$ )	32	64	128	256	512	1024	2048	4096
65,536 ( $2^{16}$ )	64	128	256	512	1024	2048	4096	8192
131,072 ( $2^{17}$ )	128	256	512	1024	2048	4096	8192	16,384
262,144 ( $2^{18}$ )	256	512	1024	2048	4096	8192	16,384	32,768
524,288 ( $2^{19}$ )	512	1024	2048	4096	8192	16,384	32,768	-
1,048,576 ( $2^{20}$ )	1024	2048	4096	8192	16,384	32,768	-	-
2,097,152 ( $2^{21}$ )	2048	4096	8192	16,384	32,768	-	-	-
4,194,304 ( $2^{22}$ )	4096	8192	16,384	32,768	-	-	-	-
8,388,608 ( $2^{23}$ )	8192	16,384	32,768	-	-	-	-	-

The Godiva scaling test suite presented in Table 5.2 was produced from all powers of two within the prescribed ranges for the three parameters, mesh size, cells per sub-domain, and number of processors. In this table, the cases comprising a row are of constant mesh size, providing the cases for a strong scaling trend. A column is comprised of cases with a constant number of cells per sub-domain, providing the cases for a weak scaling trend. Note that the mesh sizes in this table

are approximate. While we attempt to produce mesh sizes that are powers of two, obtaining meshes of an exact prescribed number of cells with GMSH is not feasible. Thus, the actual mesh sizes are not exact powers of two. The actual mesh sizes are displayed in Table 5.3. Additionally, the number of cells per sub-domain are approximate, both due to the difference in actual mesh size from target mesh size as well as the fact that the sub-domains produced by METIS for a given mesh are of slightly varying sizes. Note that for readability, in the text we refer to mesh sizes and sub-domain sizes by their approximate power of two size. The actual sizes are used in the plots, though. In addition to the mesh sizes, Table 5.3 presents the percentages of the physical Godiva problem volume that are modeled by each mesh.

Table 5.3: Target and actual mesh sizes and percent of physical volume modeled for Godiva scaling test suite

Target Mesh Size	Actual Mesh Size	Relative Difference	Percent of Volume
8 ( $2^3$ )	6	-25.00%	77.94%
16 ( $2^4$ )	19	18.75%	77.94%
32 ( $2^5$ )	34	6.25%	91.25%
64 ( $2^6$ )	59	-7.81%	94.67%
128 ( $2^7$ )	130	1.56%	96.85%
256 ( $2^8$ )	258	0.78%	98.13%
512 ( $2^9$ )	495	-3.32%	98.68%
1024 ( $2^{10}$ )	976	-4.69%	99.27%
2048 ( $2^{11}$ )	2149	4.93%	99.60%
4096 ( $2^{12}$ )	3977	-2.91%	99.71%
8192 ( $2^{13}$ )	7970	-2.71%	99.82%
16,384 ( $2^{14}$ )	15,956	-2.61%	99.88%
32,768 ( $2^{15}$ )	31,767	-3.05%	99.92%
65,536 ( $2^{16}$ )	63,523	-3.07%	99.95%
131,072 ( $2^{17}$ )	129,722	-1.03%	99.96%
262,144 ( $2^{18}$ )	254,121	-3.06%	99.97%
524,288 ( $2^{19}$ )	509,827	-2.76%	99.98%
1,048,576 ( $2^{20}$ )	1,022,111	-2.52%	99.98%
2,097,152 ( $2^{21}$ )	2,124,877	1.32%	99.99%
4,194,304 ( $2^{22}$ )	4,128,322	-1.57%	99.99%
8,388,608 ( $2^{23}$ )	8,249,741	-1.66%	99.99%

The Godiva geometric configuration is a homogeneous sphere of radius 8.741 cm. Computationally, this is modeled as one octant of the sphere with three reflective boundary conditions on the flat surfaces. We provide three sample Godiva meshes from our testing suite in Figs. 5.5 - 5.7. The visualization tool used to generate all mesh illustrations in this chapter is ParaView. [73]

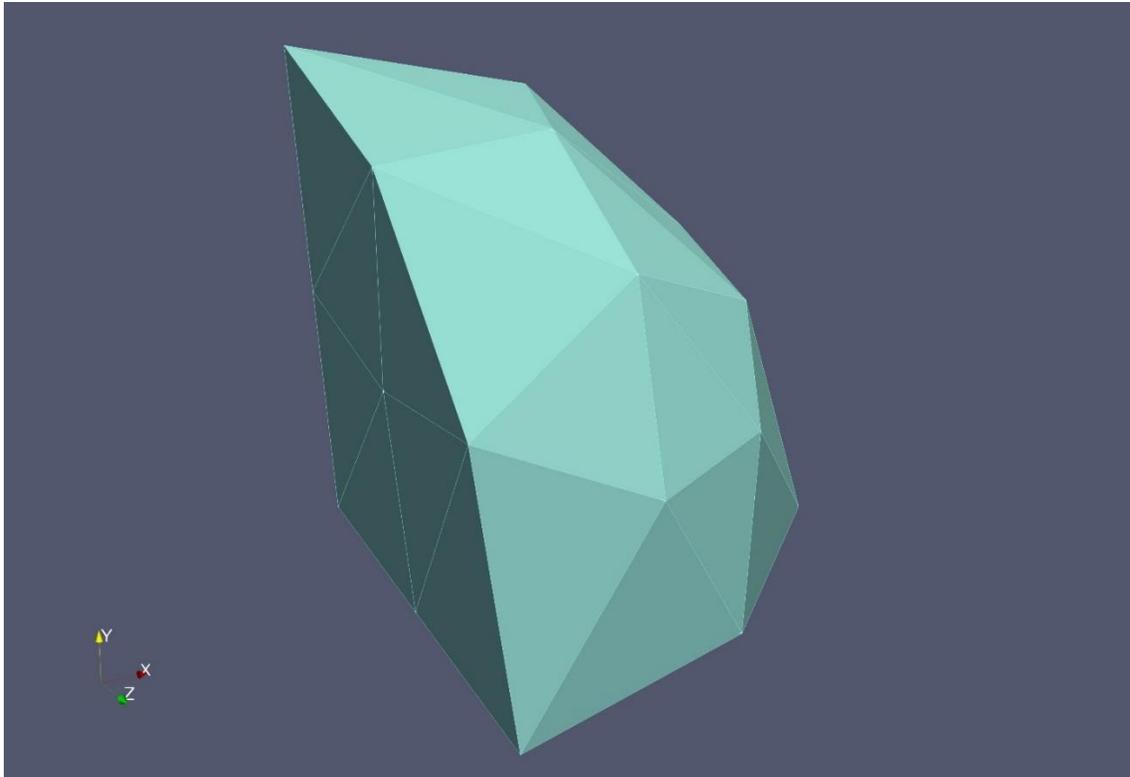


Figure 5.5: Godiva tetrahedral mesh with ~32 cells

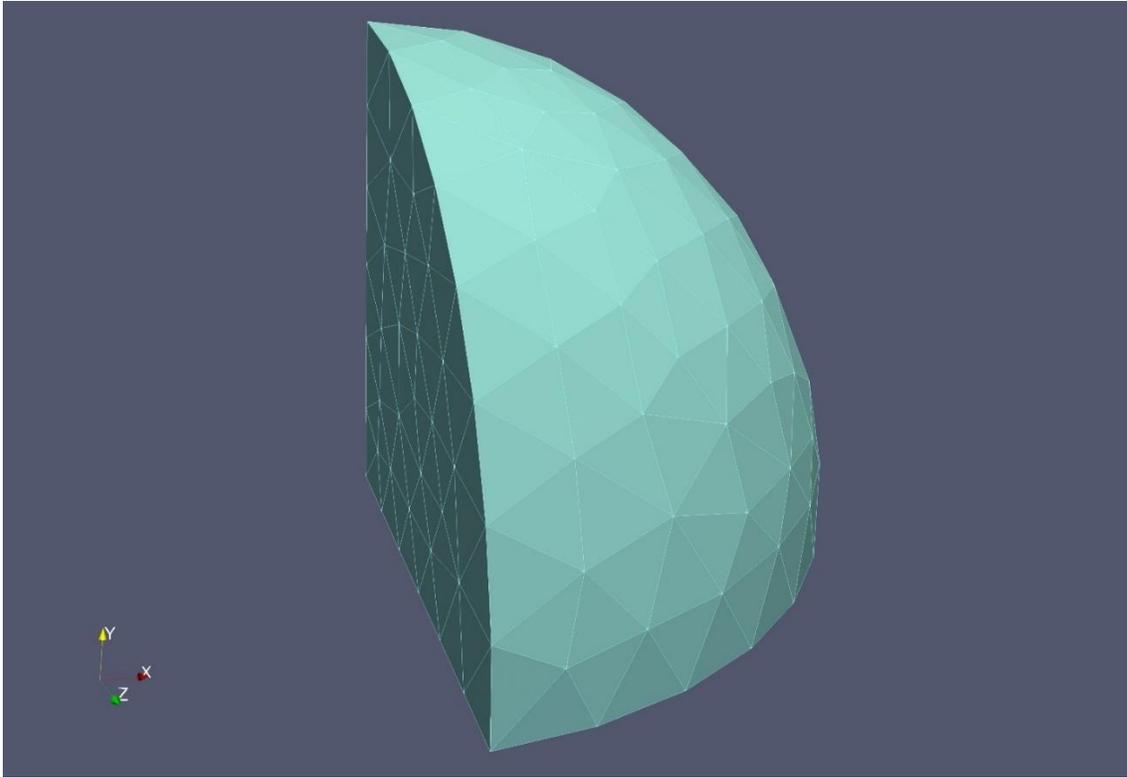


Figure 5.6: Godiva tetrahedral mesh with ~512 cells

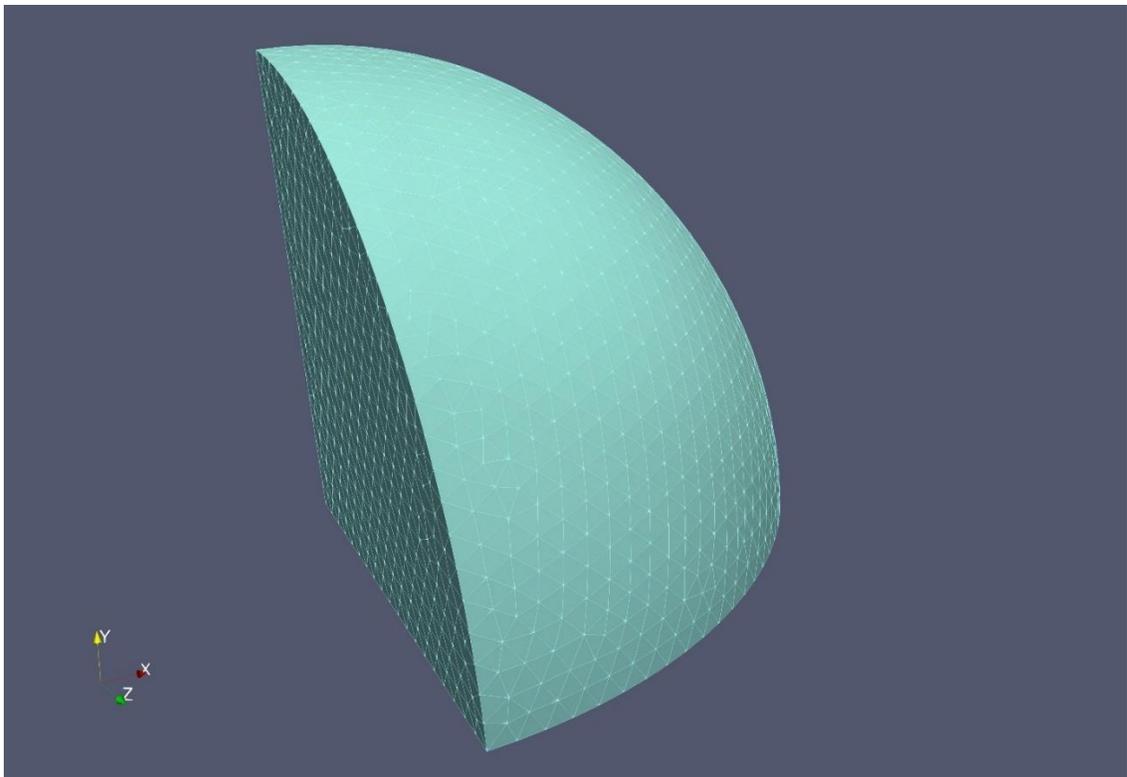


Figure 5.7: Godiva tetrahedral mesh with ~16,384 cells

These meshes present an important detail of our Godiva scaling tests. The distinction between two meshes of different size is not in the physical size of the problem, but in the respective levels of refinement. Consequently, as the mesh size increases, the optical thickness of its constituent cells decreases. As a result, we can predict that the degradation of iterative performance associated with *PBJ* methods in optically thin cells will cause an increase in the number of iterations as processor and cell counts are increased proportionately with weak scaling. While *PBJ* strong scaling with our definition of work is expected to yield an increase in iterations as processors are added due to the decreased number of cells per sub-domain, this effect is expected to be exacerbated by the decreased cell size of larger meshes. As will be discussed in the subsequent sub-section, C5G7 will be set up in a manner such that the increase in cells is due to increased problem volume, rather than mesh refinement. This allows us to have one scaling test problem that is greatly impacted by the iterative slowdown *PBJ* methods experience with mesh refinement and one in which the iterative properties of *PBJ* are less impactful. The current test problem under discussion, Godiva, is of the former type.

Samples of partitioned ~16,384 cell Godiva meshes from our test suite are presented in Figs. 5.8 - 5.10. In these figures, each color represents a different sub-domain, with the mesh overlain. Note that due to the large number of sub-domains, some sub-domains have very similar colors, but are clearly separated by other sub-domains.

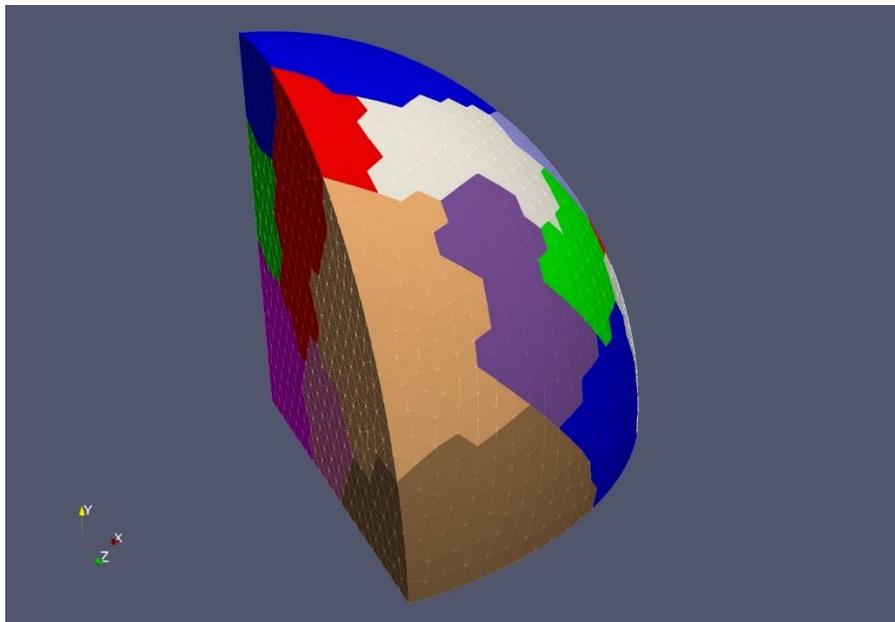


Figure 5.8: Godiva ~16,384 cell mesh, partitioned over 16 sub-domains

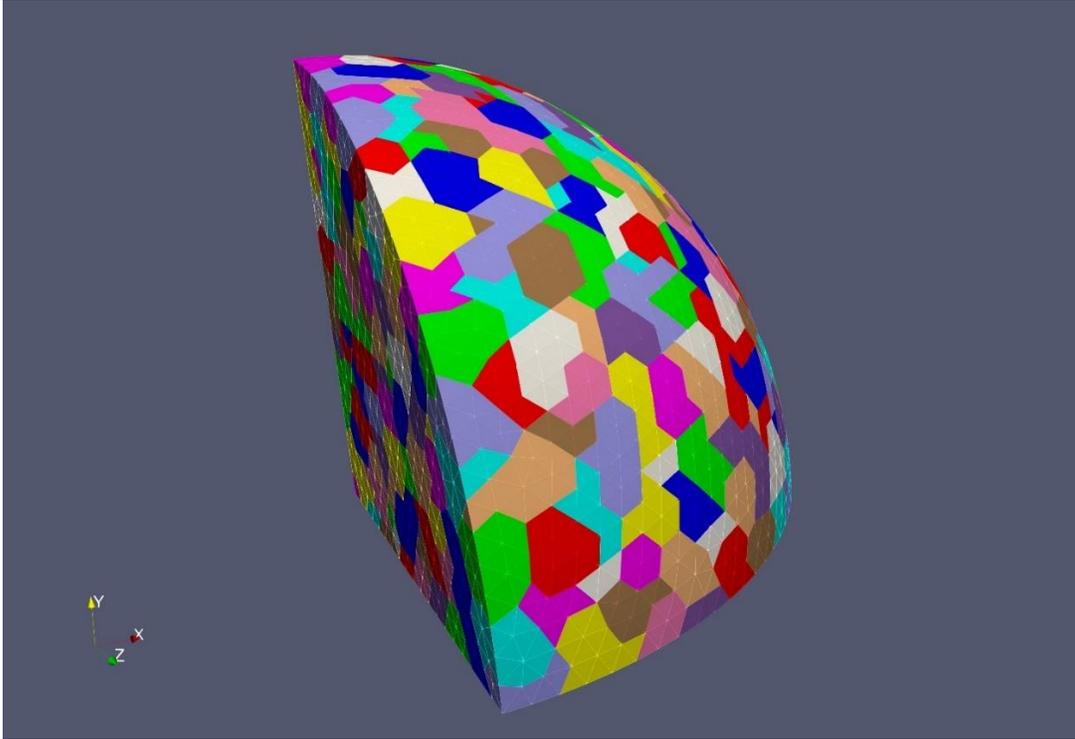


Figure 5.9: Godiva ~16,384 cell mesh, partitioned over 512 sub-domains

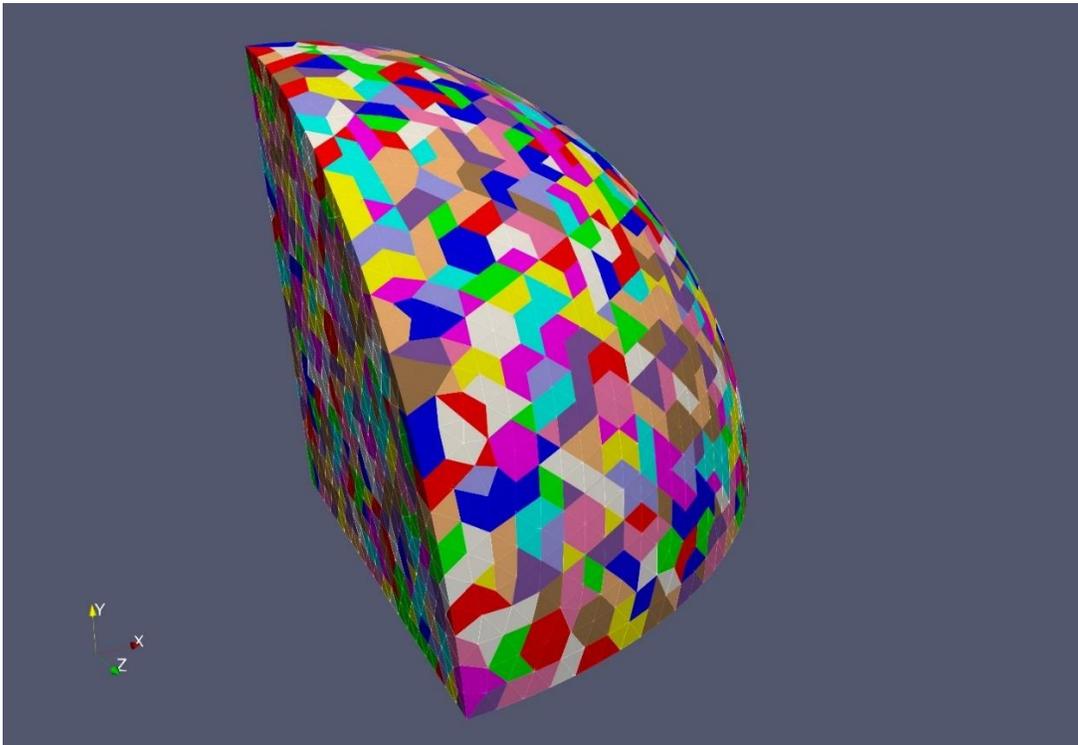


Figure 5.10: Godiva ~16,384 cell mesh, partitioned over 2048 sub-domains

With meshes generated and partitioned for each case in the Godiva test suite, all cases listed in Table 5.2 are executed on Sawtooth. Before presenting the measured performance data, we briefly discuss verification procedures for the implementation of *PBJ-ITMM* and *IPBJ* in THOR, as well as the Godiva problem inputs. For verification purposes, we present the  $k$ -eigenvalue produced by THOR for Godiva as a function of mesh size in Fig. 5.11. The progression of  $k$  towards the benchmark value near 1 (0.996878 for the largest mesh) verifies the integrity of the THOR inputs generated for this testing suite, with the trend also correlating to the percentages of the Godiva problem modeled, displayed in Table 5.3. Additionally, the fact that the values of  $k$  produced for a given mesh size are identical, regardless of the method or number of partitions used, increases confidence in our *PBJ* method implementations. For additional verification, we compared our results for certain meshes to those produced by the standard serial version of the THOR code, verifying that the results are equivalent to within the convergence criterion.

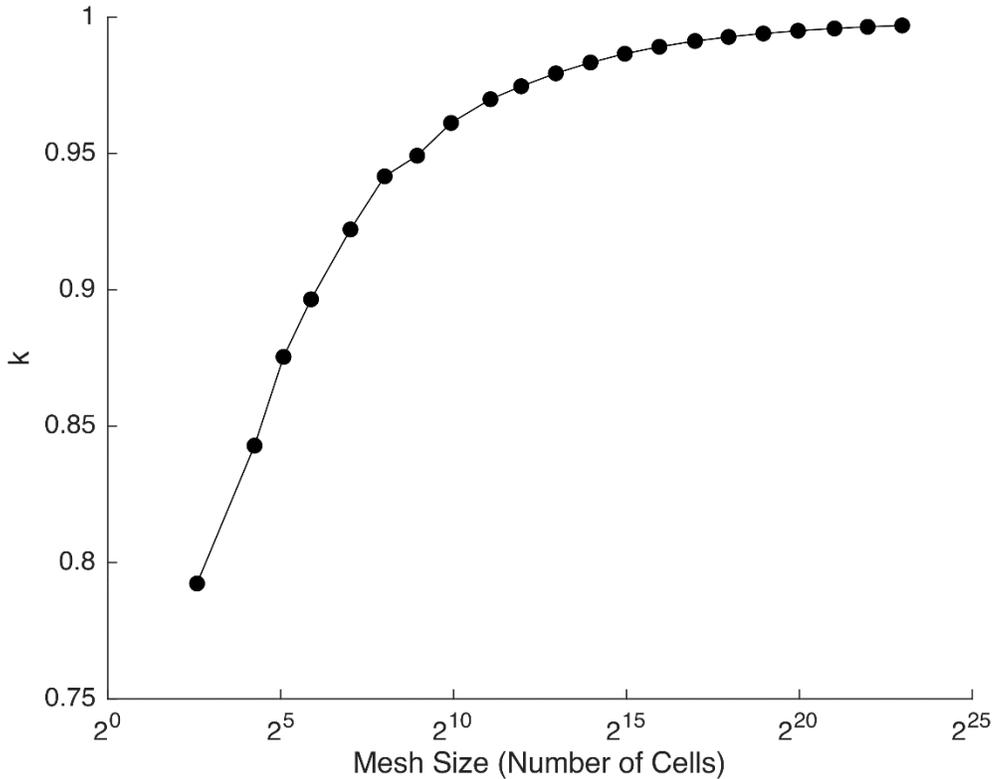


Figure 5.11:  $k$ -eigenvalues for Godiva produced by parallel THOR vs. mesh size

With our *PBJ* implementations and Godiva problem inputs verified, the first timing metric we consider is the time required by *GFIC* to construct the *ITMM* matrices for a given sub-domain size, as *GFIC* is the novel algorithm developed in this work which allows these matrices to be constructed in parallel on unstructured grids. The construction times for each Godiva case are displayed in Fig. 5.12. Note that each point color in this plot represents a different global mesh size. We omit a legend from this plot, however, since the global mesh size should not affect the construction time, as this is a local process. We include the data points from all mesh sizes to demonstrate that the observed construction time follows this expectation and is only dependent on sub-domain size.

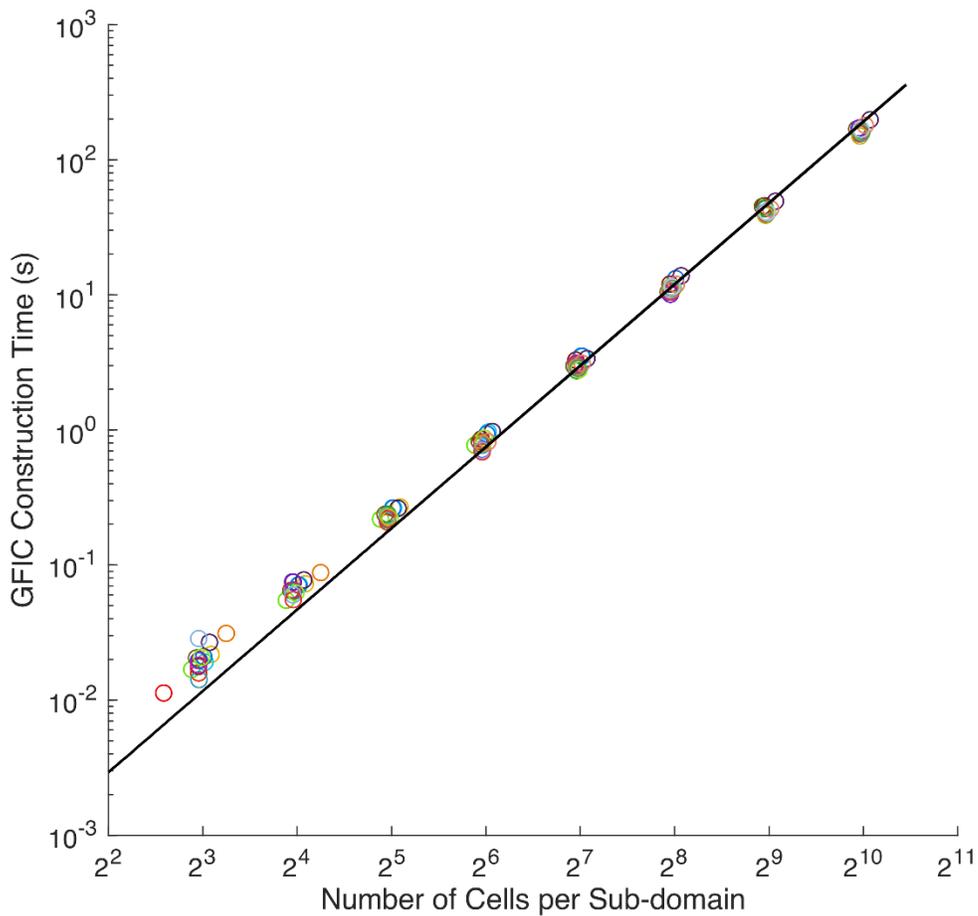


Figure 5.12: Time consumed by *GFIC* to construct *ITMM* matrices for Godiva in THOR (points) and leading order of theoretical construction time, Eq. (3.88) (line)

From this plot, it is confirmed that *GFIC*'s construction time follows the same leading order as Eq. (3.88), which was derived based on Cartesian grids, but with full expectation that it would be applicable to unstructured grids as well. Additionally, the construction time is very consistent across global mesh sizes. This was also expected, as the construction process is purely localized on each processor. Finally, this trend demonstrates that *GFIC* constructs the *ITMM* matrices expeditiously. For the largest sub-domain size ( $\sim 1024$  cells), the construction time is only a few minutes for Godiva, which is a 6-group problem. This sub-domain size is well above what is expected to be ideal, and at this size, construction time is still a reasonable one-time setup cost.

With *GFIC*'s construction cost found to be both reasonable and consistent with expectation, we proceed to the strong scaling results for Godiva. With our scaling tests we report, for both *PBJ-ITMM* and *IPBJ*, the number of required iterations (both outer and inner) and multiple execution times associated with the solution process. These times include the total solution time consumed as well as the times associated with the three constituents of the solution, pre-process, computation, and communication. Total solution time is defined as the time required to reach observable convergence after the initial I/O and problem setup has completed. Pre-process time is all time associated with the solution that precedes the start of the iterative process. This includes *GFIC* construction and matrix factorization (for *PBJ-ITMM*), communication instructions generation, transmission of communication instructions between processors, and the generation of pack and unpack instructions. Computation time is then defined as all time during the iterative process that is not associated with MPI communication or pack/unpack processes. These communication and pack/unpack processes comprise the communication time. The sum of these three constituent times is equal to the total solution time. We present the times consumed by these constituent processes to expose which process dominates the solution time, to what degree it dominates, and how this varies across cases.

To create a strong scaling trend, we plot the measured time for one row from the Godiva test suite, Table 5.2. To best present the important findings, we consider strong scaling trends from a mesh size of  $\sim 32,768$  to  $\sim 262,144$  cells, the required iterations and solution times for which are presented in Figs. 5.13 - 5.20. These selected mesh sizes limit the trends to those which span the studied range of sub-domain sizes, neglecting smaller meshes due to the associated small processor counts. Full iteration and timing data from the Godiva testing suite, Table 5.2, is available in Appendix C.

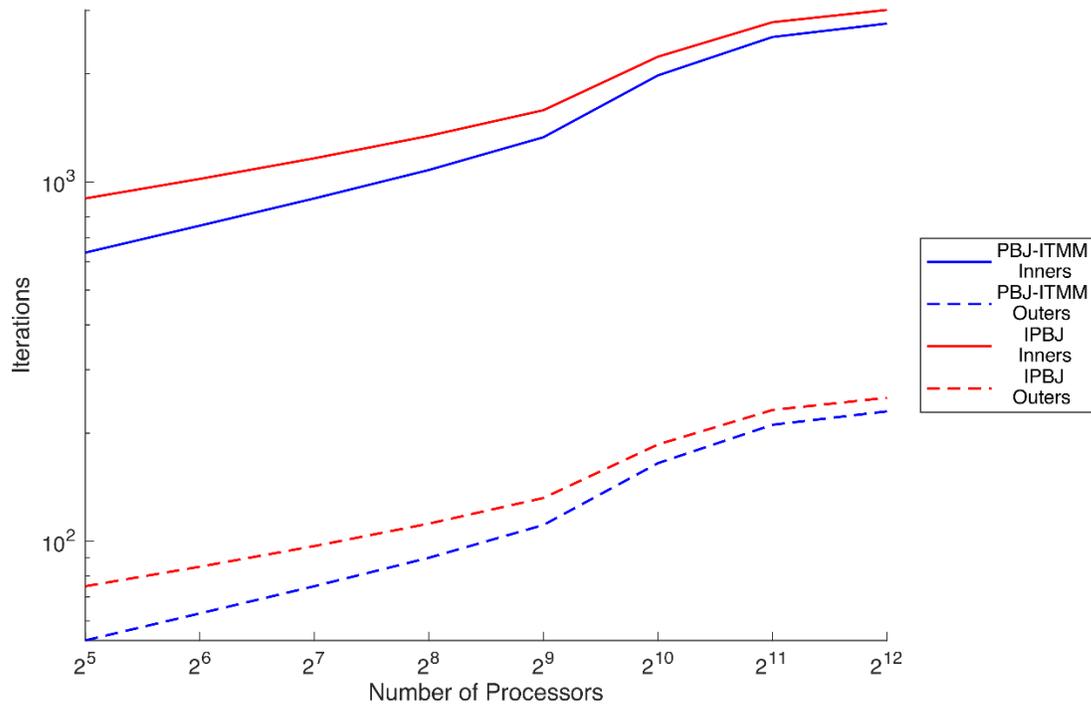


Figure 5.13: Iteration strong scaling for Godiva: ~32,768 cell mesh

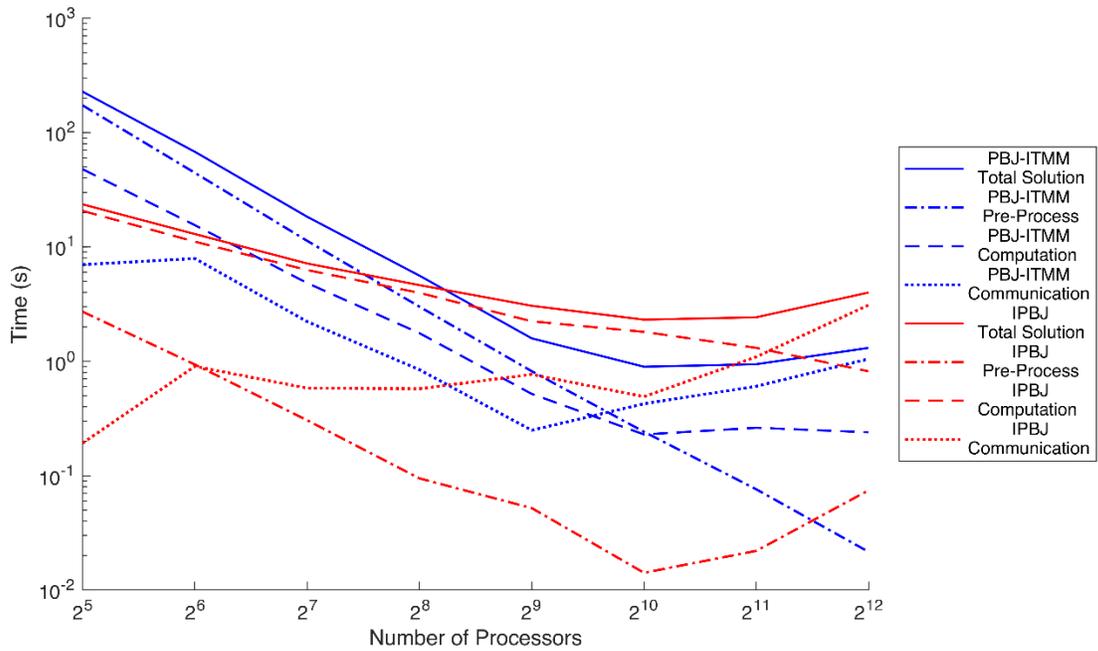


Figure 5.14: Solution time strong scaling for Godiva: ~32,768 cell mesh

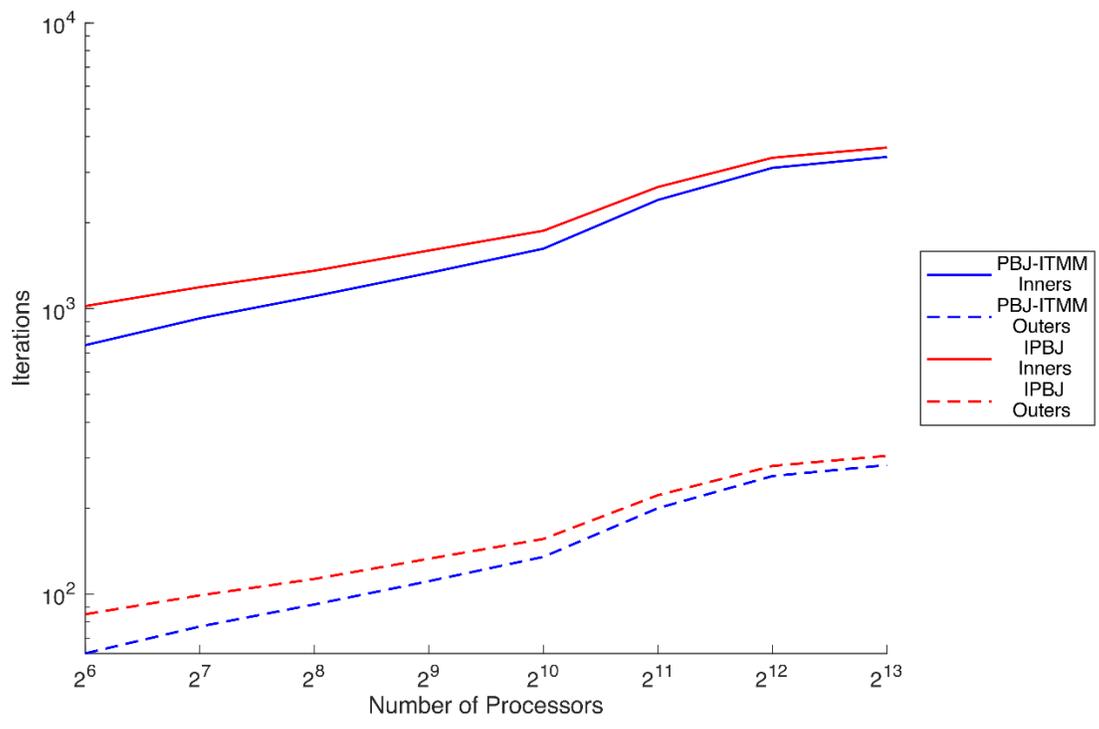


Figure 5.15: Iteration strong scaling for Godiva: ~65,536 cell mesh

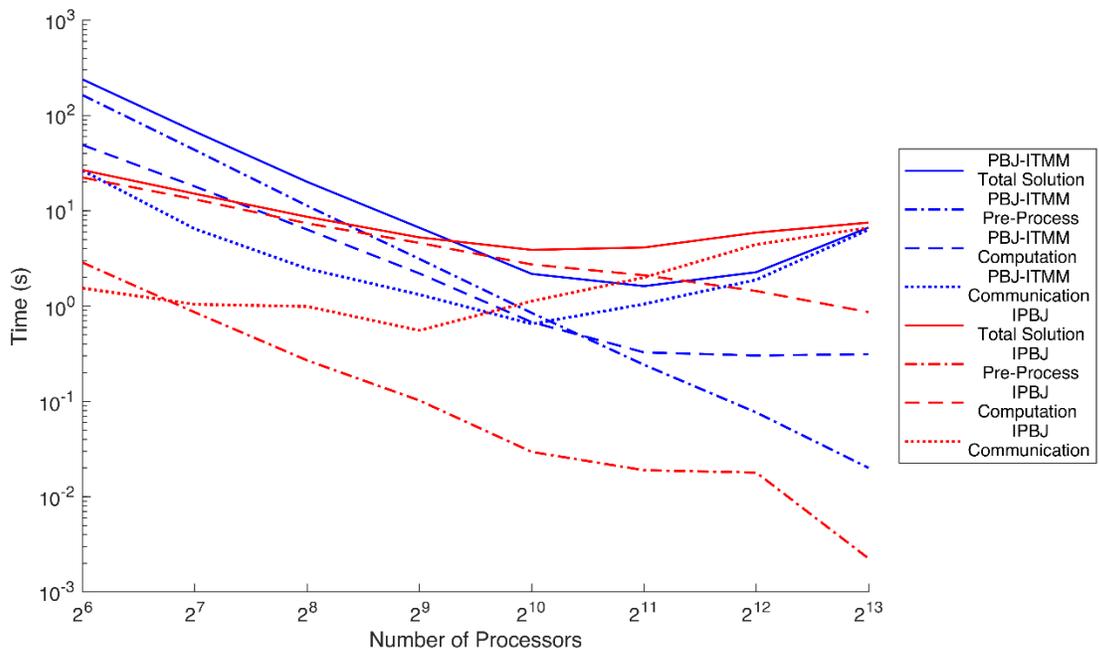


Figure 5.16: Solution time strong scaling for Godiva: ~65,536 cell mesh

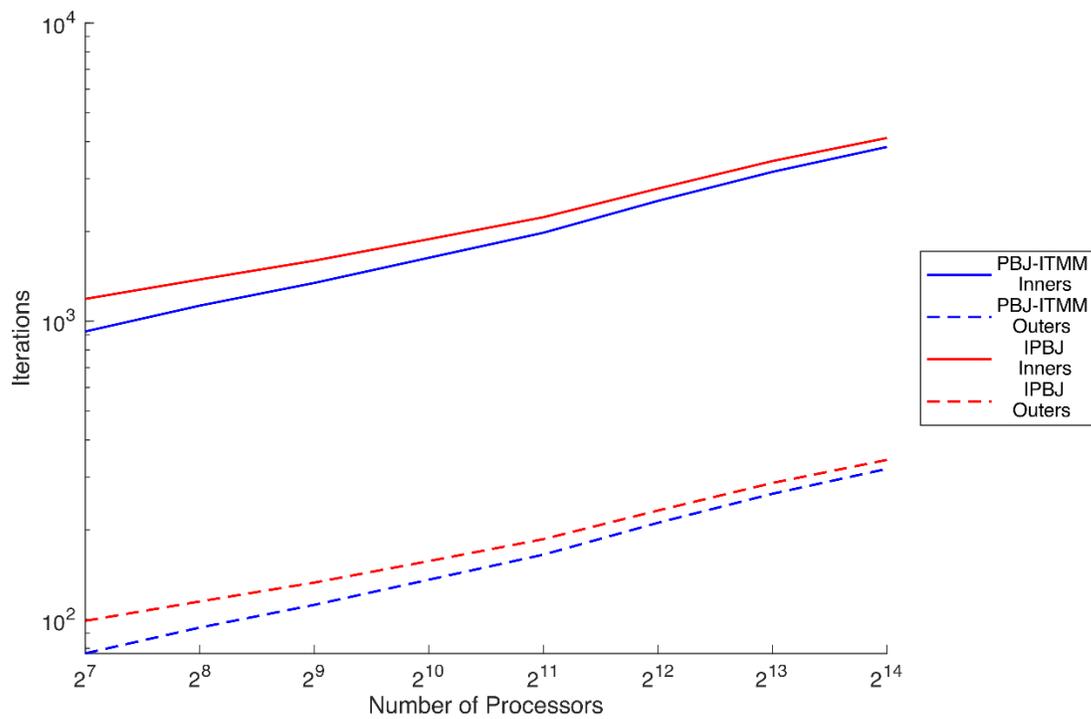


Figure 5.17: Iteration strong scaling for Godiva: ~131,072 cell mesh

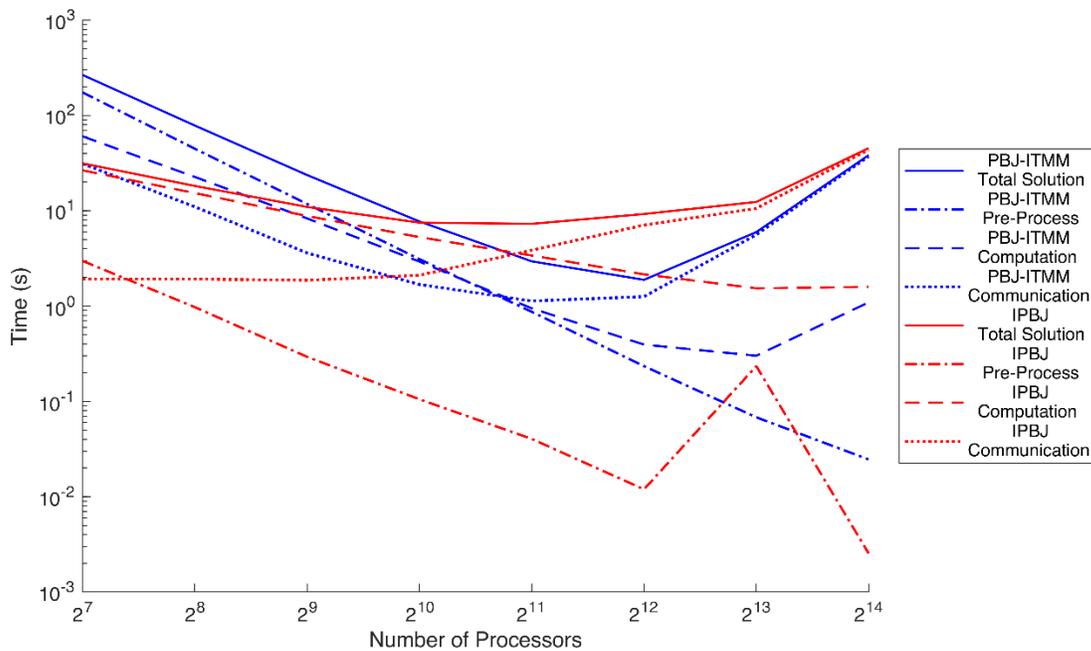


Figure 5.18: Solution time strong scaling for Godiva: ~131,072 cell mesh

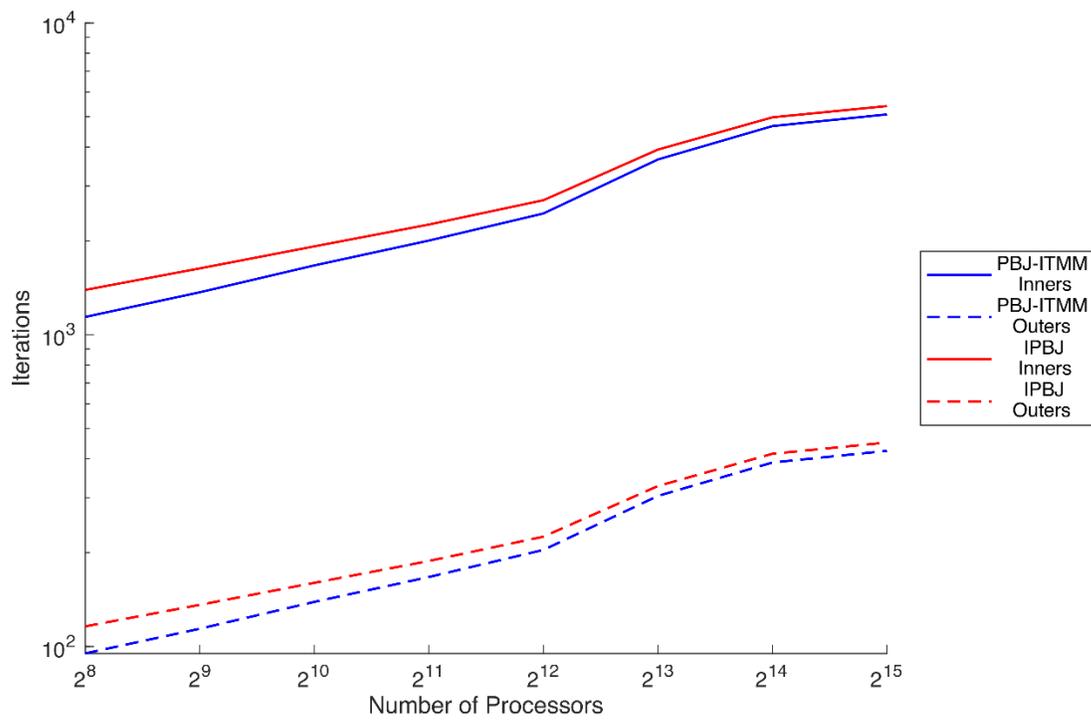


Figure 5.19: Iteration strong scaling for Godiva: ~262,144 cell mesh

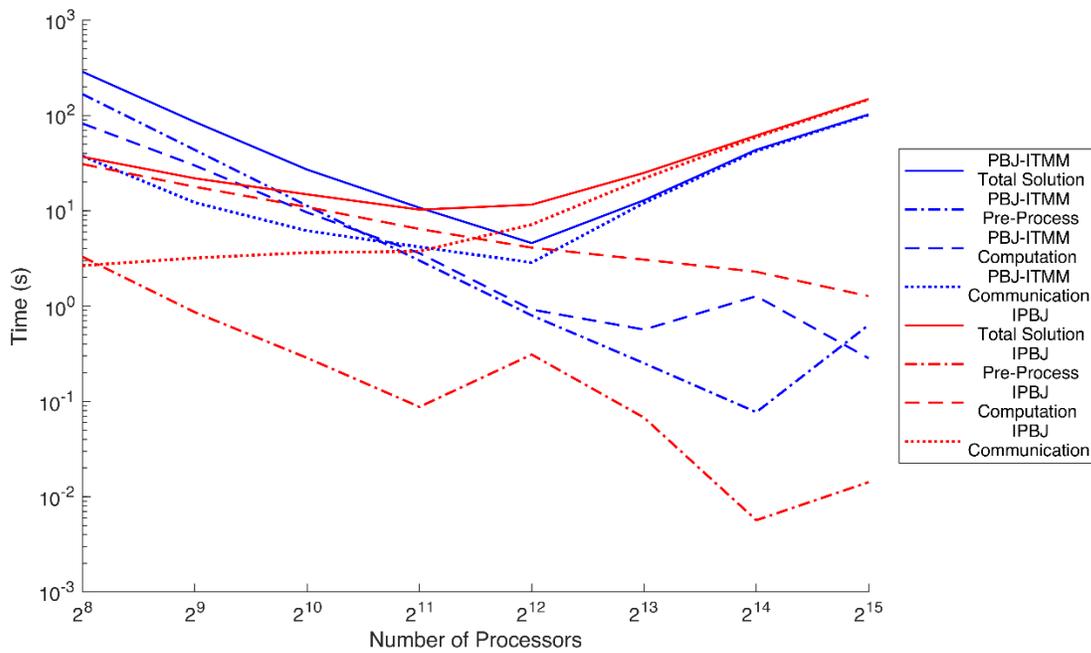


Figure 5.20: Solution time strong scaling for Godiva: ~262,144 cell mesh

Considering Figs. 5.13, 5.15, 5.17, and 5.19, the Godiva strong scaling iteration trends are consistent with expectation. Firstly, the number of iterations, both outer and inner, consumed by *IPBJ* is greater than that of *PBJ-ITMM* for all cases, as has consistently been predicted and observed in all previous work. Additionally, for a given mesh size, the required number of iterations for both methods increases as the number of processors increases. This is an expected result for any strong scaling trend using *PBJ*-type methods, as the decreasing number of cells per sub-domain increases the average number of sub-domain interfaces crossed by particles. Comparing *PBJ-ITMM* and *IPBJ* from the standpoint of iterations, as the number of processors increases for a given mesh size, the relative difference between the iteration counts for the two methods becomes smaller, as the decreased sub-domain size increases the iterative degradation from *PBJ*'s asynchronicity relative to that of the lagged scattering source in *IPBJ*.

Proceeding to the strong scaling of solution time, Figs. 5.14, 5.16, 5.18, and 5.20, we analyze the trends of how the fraction of the constituent execution times vary. For *PBJ-ITMM*, in general, at low processor counts (i.e. large numbers of cells per sub-domain) the pre-process time heavily dominates computation and communication times, comprising the majority of the total solution time in many cases. For a given mesh size though, as the number of processors increases, communication and/or computation comprise larger percentages of the total solution time, as the cost of *GFIC* falls rapidly with the decreasing number of cells per sub-domain. The relative proportions of computation and communication times as processor count increases depend on the mesh size. Generally, bigger meshes result in communication becoming far larger than computation with increasing processor count. This is because the number of processors for large meshes is larger for the same number of cells per sub-domain and normally the communication time increases with the number of participating processors. This effect is discussed in more detail subsequently in the weak scaling results.

One final effect observed in the *PBJ-ITMM* timing results for the strong scaling tests is the eventual increase in solution time, i.e. slowdown, as processor count is increased. In all mesh sizes presented, there is a point at which increasing the number of processors actually increases the total solution time. At the processor counts where this occurs, the pre-process time is mostly insignificant compared to the communication and/or computation time. At a certain point, the negative effects of increased processor count (increase in required iterations and non-ideal communication cost scaling) outweigh the benefits (reduced grind time and number of messages

per sub-domains) and the addition of processors actually increases the total solution time. This result is predictable and undamaging to *PBJ-ITMM*'s application for massively parallel solution on unstructured grids. *PBJ* methods were not pursued for their ability to gratuitously parallelize problems of moderate size, but for their scalability to large-scale problems. Due to their asynchronicity, it is expected that there is a minimum sub-domain size, below which the increase in number of iterations outweighs the diminishing iteration cost.

Continuing our analysis of the strong scaling timing data, Figs. 5.14, 5.16, 5.18, and 5.20, we consider *IPBJ*'s performance. For all cases, the pre-process time for *IPBJ* is insignificant, with one of the other time components always greatly exceeding it. This is expected, as the pre-process for *IPBJ* is comprised only of the generation of communication instructions, requiring none of the kernel calculations performed by *GFIC*. Consequently, the trends for *IPBJ* are somewhat simpler than for *PBJ-ITMM*, as the main effect of pertinence is the varying proportions of communication and computation times. Generally, for a given mesh size, *IPBJ*'s solution time is dominated by computation at low processor counts and by communication at high processor counts. Additionally, the trends are very consistent, with computation and communication time almost continuously decreasing and increasing, respectively, with increased processor count.

Comparing the timing trends for *PBJ-ITMM* and *IPBJ*, the primary observation is that the total solution time for *IPBJ* is shorter with large sub-domains and *PBJ-ITMM*'s solution time is shorter with small sub-domains. While this is an expected behavior, the sub-domain size at which *PBJ-ITMM* becomes faster is problem-dependent. The transition point where *PBJ-ITMM*'s total solution time becomes smaller than *IPBJ*'s consistently occurs at  $\sim 128$  cells per sub-domain for Godiva. Observation of this transition point confirms that  $\sim 1024$  cells per sub-domain was a large enough upper bound on sub-domain size to fully encompass the range over which *PBJ-ITMM* executes faster than *IPBJ*.

Another observation comparing the two methods is contradictory to previous expectation. When sub-domains are large, the communication time for *PBJ-ITMM* is actually longer than for *IPBJ*, despite having the same per-iteration communication requirements and fewer required iterations. This is a consequence of the exact process used to time the communication phase. All timing data is collected on the root processor. The starting time for the communication phase is collected directly before this processor begins packing its messages. The ending time, however, must occur after the `MPI_WAITALL` that follows the send and receive operations, as there would

otherwise be no guarantee that the requested messages had been received. Because the sub-domains are of slightly varying sizes, processors enter the communication phase at different times, due to slightly varying computational costs associated with the iterative solution. There is consequently a portion of the communication phase that is spent waiting for slightly larger sub-domains to complete their iterative solution. While this occurs with both *PBJ-ITMM* and *IPBJ*, it is more observable with *PBJ-ITMM* due to the non-linear scaling of the iterative solution time with respect to sub-domain size. This, coupled with the small communication costs due to the small processor counts when sub-domains are large, results in the observed communication cost for *PBJ-ITMM* being larger than *IPBJ*'s. As processor count increases, however, *PBJ-ITMM* ends up requiring shorter communication times than *IPBJ*, as predicted.

The final observation is that the uptick in the solution time as sub-domains become very small is more prevalent with *PBJ-ITMM* than with *IPBJ*. The effect that is likely responsible for this is the decreasing relative difference in the number of required iterations as processor count increases. The resulting relative increase in communication cost as processor count increases is therefore larger with *PBJ-ITMM*.

The strong scaling results for Godiva are consistent with expectation. *PBJ-ITMM* is the faster solution method with sub-domains up to a certain size, after which *IPBJ*'s linearly scaling per-iteration solution time with respect to the number of cells per sub-domain outweighs its increased number of required iterations. Additionally, as processor count increases, the dominant process shifts from *GFIC* (for *PBJ-ITMM*) or computation (for *IPBJ*) to communication. Finally, due to the asynchronicity of *PBJ* methods, these methods are poorly suited to excessively parallelize a small problem, as the increase in number of iterations from sub-domains with few cells outweighs the reduced per-iteration cost.

Continuing our scaling results for Godiva, we proceed to the weak scaling tests. A weak scaling study keeps the number of cells per sub-domain constant, increasing mesh size proportionately with the number of processors. To create one weak scaling trend, we plot the results for one column of Table 5.2, with the resulting trends presented in Figs. 5.21 - 5.28. Recall that the primary finding of the strong scaling tests was the range over which *PBJ-ITMM* is viable for this problem. This range is from ~32 to ~128 cells per sub-domain. The lower bound is determined by the point at which adding processors consistently results in increased total solution time and the upper bound is determined by the point at which *IPBJ* becomes faster than *PBJ-*

*ITMM*. We present only the weak scaling trends over this range of sub-domain sizes because THOR contains the first *PBJ-ITMM* implementation on unstructured grids via *GFIC* and we wish to study its performance in a practical capacity. The ~256 cells per sub-domain weak scaling trend is additionally included as it contains the largest problem tested in the Godiva suite. As was the case with strong scaling, we present the number of iterations (outer and inner) and the total solution time along with its three components, pre-process, computation, and communication times.

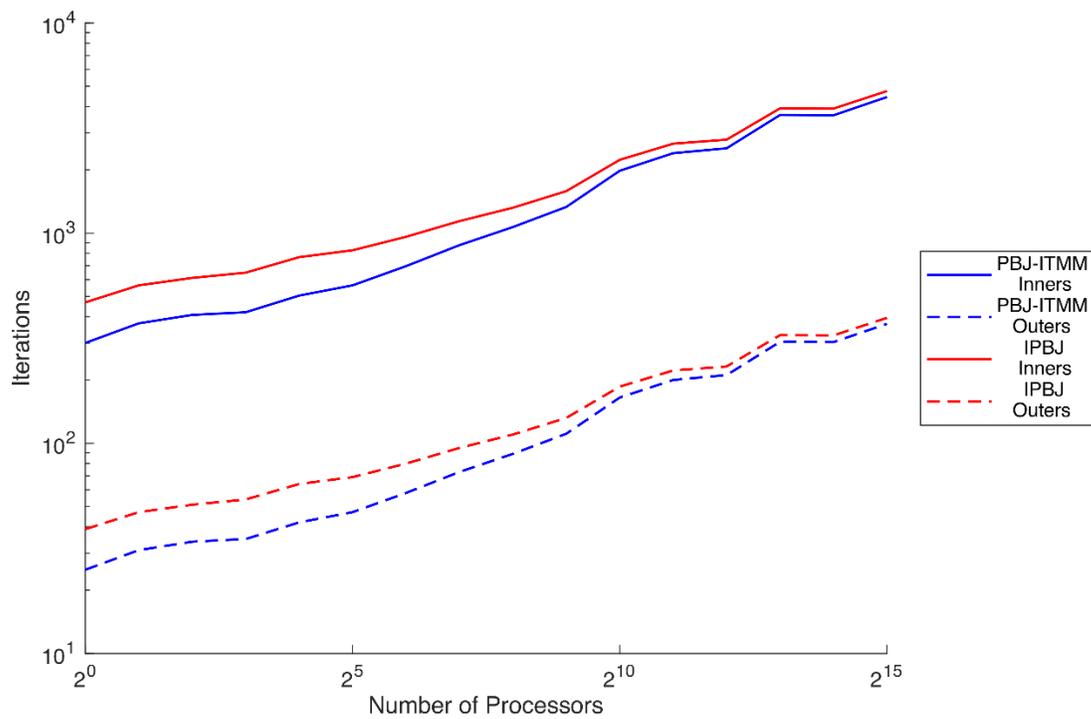


Figure 5.21: Iteration weak scaling for Godiva: ~32 cells per sub-domain (processor)

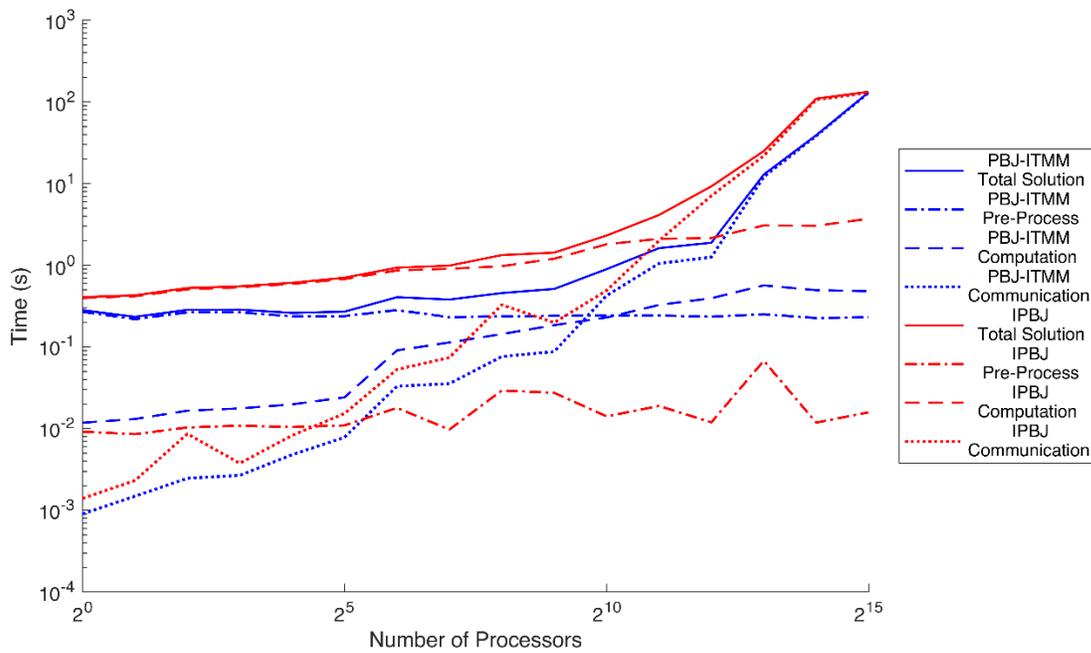


Figure 5.22: Solution time weak scaling for Godiva: ~32 cells per sub-domain (processor)

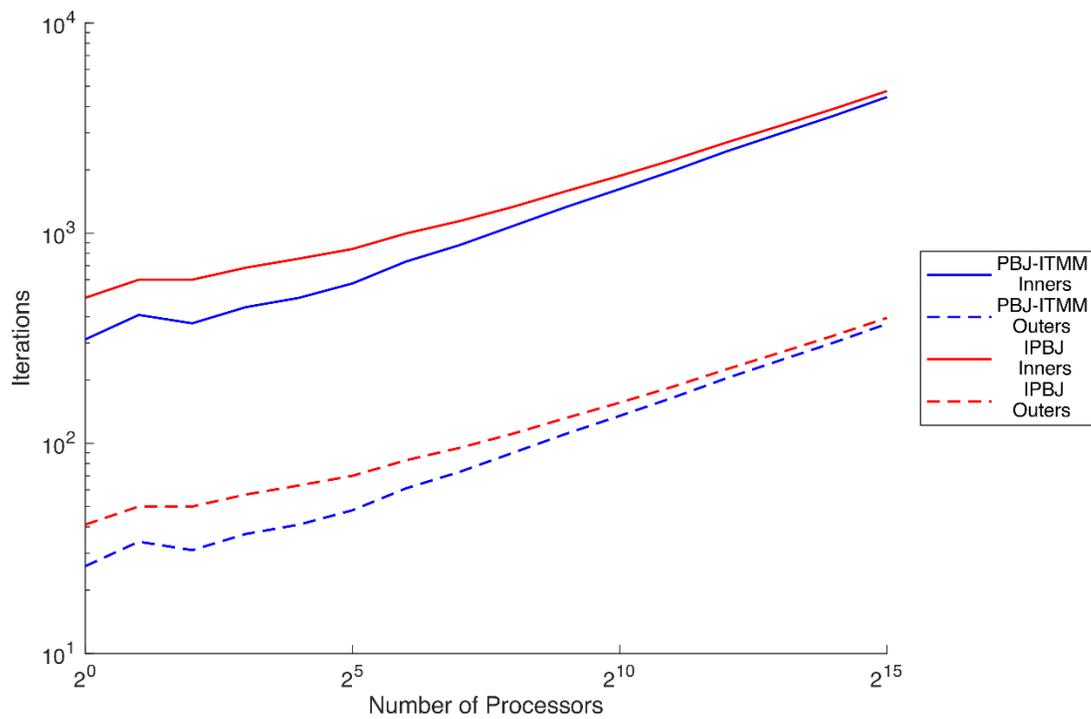


Figure 5.23: Iteration weak scaling for Godiva: ~64 cells per sub-domain (processor)

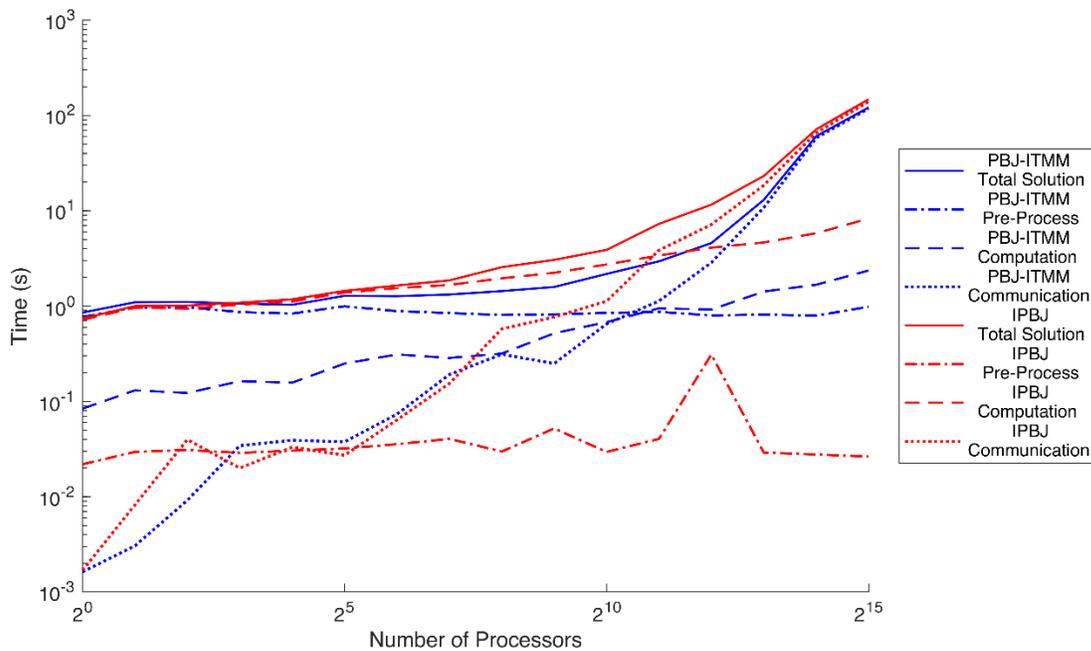


Figure 5.24: Solution time weak scaling for Godiva: ~64 cells per sub-domain (processor)

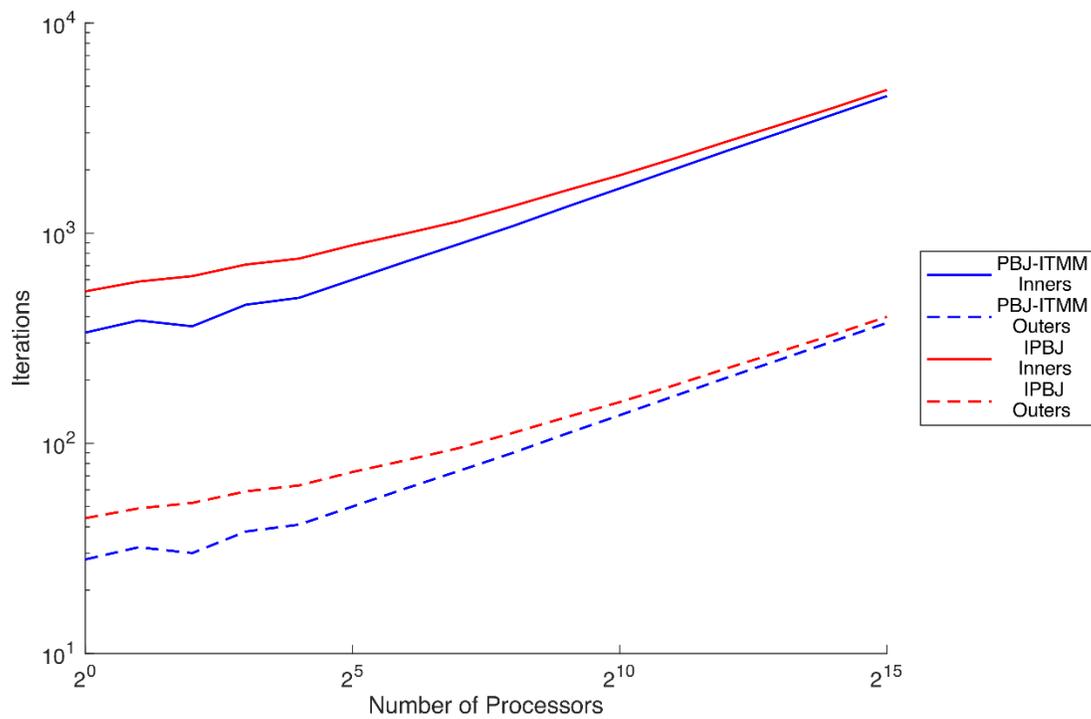


Figure 5.25: Iteration weak scaling for Godiva: ~128 cells per sub-domain (processor)

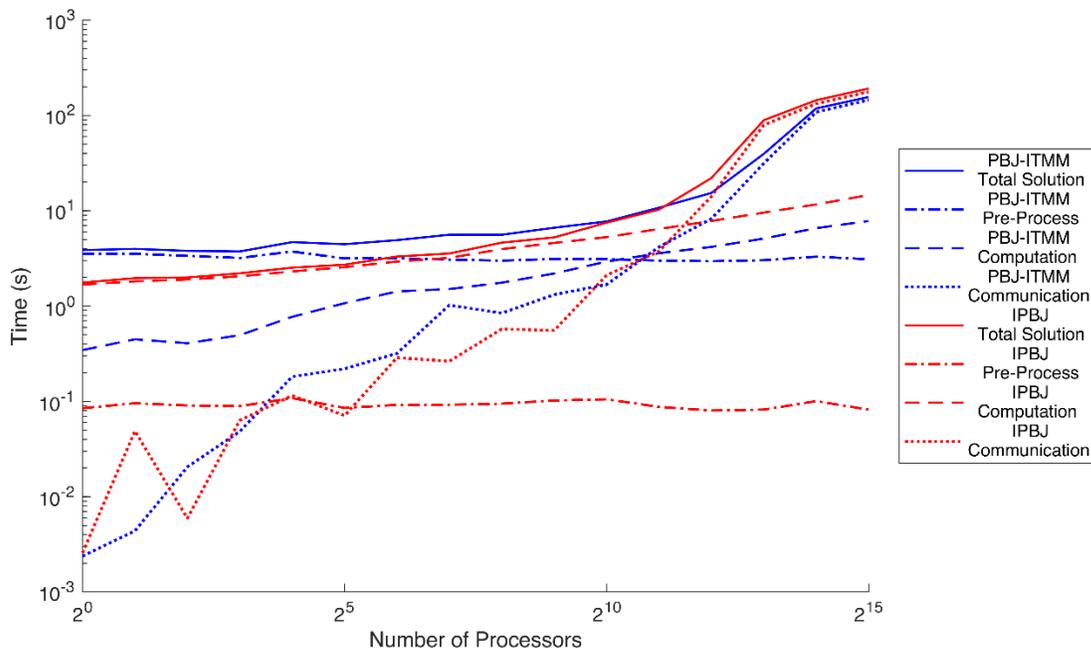


Figure 5.26: Solution time weak scaling for Godiva: ~128 cells per sub-domain (processor)

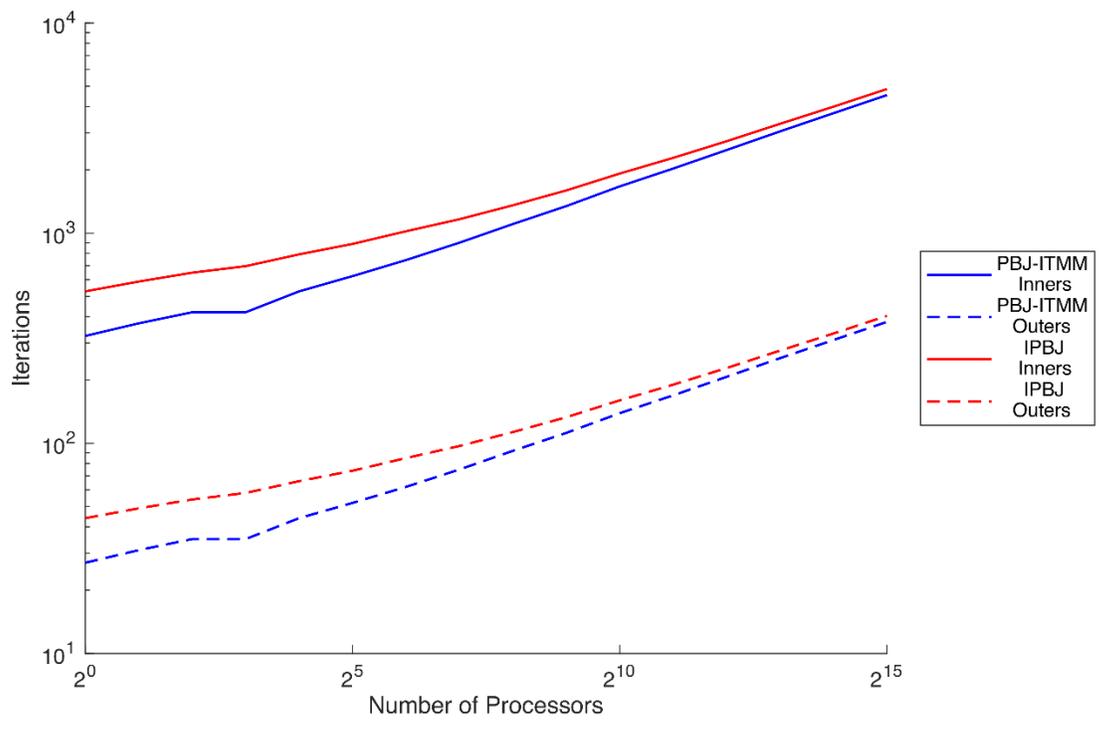


Figure 5.27: Iteration weak scaling for Godiva: ~256 cells per sub-domain (processor)

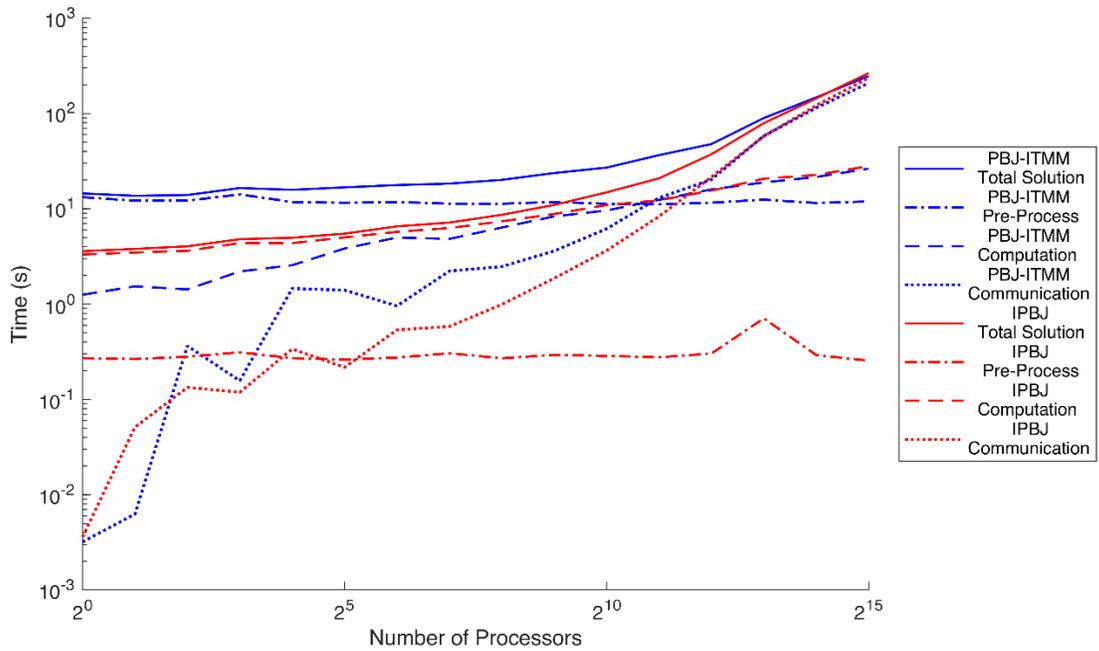


Figure 5.28: Solution time weak scaling for Godiva: ~256 cells per sub-domain (processor)

Analyzing the iteration weak scaling results for Godiva, Figs. 5.21, 5.23, 5.25, and 5.27, we again observe *PBJ-ITMM* to always converge in fewer iterations than *IPBJ* noting that the number of required iterations for both methods increases as the number of processors, hence sub-domains, increases. As was the case in the strong scaling results, the relative difference in the number of iterations required by the two methods decreases as the processor count increases. The physical size of the Godiva problem does not change with mesh size. Instead, increased mesh size is achieved via mesh refinement, which reduces the optical thickness of the cells on average, degrading the iterative performance of *PBJ* methods. From the iterative weak scaling trends, it is clear that for a mesh this heavily refined, the lagged scattering source in *IPBJ* has little additional effect on iterative convergence rate over the effect of the asynchronicity. The large increase in the number of iterations as the processor count rises was an intentional feature of this scaling test suite, as the increased iteration counts affect the execution time trends, the analysis of which follows.

Analyzing the total solution time weak scaling trends for Godiva, Figs. 5.22, 5.24, 5.26, and 5.28, the effect of this refinement is visible, with the total solution times of both *PBJ-ITMM* and *IPBJ* increasing significantly with the number of processors for a given number of cells per sub-domain. For *PBJ-ITMM*, pre-process is the largest contribution at low processor counts and communication in the largest contribution at high processor counts, with a small range, if any, in the middle over which computation is the largest contribution. These results demonstrate the effectiveness of *GFIC*, with the pre-process time effectively constant with respect to sub-domain size and comprising an insignificant fraction of total solution time at high processor counts. With *GFIC*'s cost being modest for sub-domain sizes within the presented range and it being a one-time setup cost that does not increase with the degradation of iterative performance or the increase in communication time associated with high processor counts, *GFIC* is expected to perform well if extended to 100,000s of processors.

The trends of *IPBJ*'s solution time and the contributions from the pre-process, computation, and communication components are similar, but also simpler due to *IPBJ*'s linear scaling of iterative solution time with respect to sub-domain size as well as its extremely short pre-process time. As previously observed in Godiva's strong scaling tests, the pre-process time for *IPBJ* is insignificant across all cases. The dominant contribution to *IPBJ*'s solution time is computation at small processor counts and communication at large processor counts.

Comparing *PBJ-ITMM* and *IPBJ*, *PBJ-ITMM* is generally observed to execute faster, except for cases using  $\sim 256$  or  $\sim 128$  (only with small processor counts) cells per sub-domain. The sub-domain sizes for the presented weak scaling trends were specifically chosen for this fact, as between  $\sim 32$  and  $\sim 128$  cell per sub-domain is considered the range over which *PBJ-ITMM* is preferable for the Godiva problem. Due to the fact that the mesh size in this testing suite was increased through spatial mesh refinement, the solution time grows with increasing processor count for a given number of cells per sub-domain, due not only to the growing communication costs intrinsically associated with increased processor counts, but also due to the increase in number of iterations as sub-domains become optically thinner. This effect was most severe, however, when a sphere of radius 8.741 cm was refined into millions of cells. In most applications, this extreme level of refinement will not typically be performed globally, but rather, only in specific areas of interest. In such scenarios, the hybrid methods developed, analyzed, and tested in Chapters 3 and 4 would likely mitigate this feature.

This feature notwithstanding, the  $\sim 8,388,608$  cell mesh with an  $S_4$  angular quadrature and six energy groups constitutes a very large system of equations, with over 49 million cell-averaged scalar fluxes, over 1.1 billion individual cell/angle/group combinations, and a total of over 5.5 billion unknowns in the short characteristics discretization method used by THOR. This number of unknowns is determined using the average number of canonical tetrahedra per cell, estimated to be approximately 3.66. [38] From Fig. 5.28, both *PBJ-ITMM* and *IPBJ* are found to solve this problem in under five minutes using 32,768 processors. This result demonstrates the viability of *PBJ-ITMM* and *IPBJ* for solution on up to this number of processors, with the preferred method depending on the number of cells per sub-domain. For this problem *PBJ-ITMM* is found to be the preferred method for up to  $\sim 128$  cells per sub-domain.

### 5.6.2: C5G7 Scaling Results

The second test problem used for strong and weak scaling tests is the C5G7 [72] benchmark, a  $4 \times 4$  grid of nuclear fuel assemblies in a checkerboard pattern of  $\text{UO}_2$  and mixed oxide (MOX) assemblies, surrounded by moderator (water). Each individual assembly is a  $17 \times 17$  grid of fuel pins. The  $4 \times 4$  grid of fuel assemblies is modeled computationally as a  $2 \times 2$  grid with two reflective boundary conditions. As with our Godiva tests, we develop minimum and

maximum values for mesh size, processor count, and number of cells per sub-domain, then create a test suite out of all combinations of powers of two that lie within the prescribed ranges. The minimum and maximum values for number of cells per sub-domain ( $\sim 8$  and  $\sim 1024$ ) and processor count (1 and 32,768) are the same as they were for Godiva, as the reasons for these values discussed in the previous section are not problem-dependent. While the maximum mesh size remains the same at  $\sim 8,388,608$  ( $2^{23}$ ), the minimum mesh size is changed to  $\sim 262,144$  ( $2^{18}$ ). This minimum mesh size was increased significantly from the value used for Godiva due to the heterogeneity of the C5G7 configuration. The testing suite for C5G7 based on these prescribed ranges is presented in Table 5.4.

Table 5.4: C5G7 scaling test suite

Target Mesh Size	Number of Processors (Sub-Domains)							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	256	512	1024	2048	4096	8192	16,384	32,768
524,288 ( $2^{19}$ )	512	1024	2048	4096	8192	16,384	32,768	-
1,048,576 ( $2^{20}$ )	1024	2048	4096	8192	16,384	32,768	-	-
2,097,152 ( $2^{21}$ )	2048	4096	8192	16,384	32,768	-	-	-
4,194,304 ( $2^{22}$ )	4096	8192	16,384	32,768	-	-	-	-
8,388,608 ( $2^{23}$ )	8192	16,384	32,768	-	-	-	-	-

Note that this is simply the bottom six rows of the Godiva test suite (Table 5.2), as only the minimum mesh size was altered between these two problems. A row or column of Table 5.4 again indicate the cases that comprise a strong or weak scaling trend, respectively. As with Godiva, the mesh sizes in this table are target values, with GMSH unable to produce meshes of an exact prescribed value. The actual mesh sizes for these target sizes is provided in Table 5.5.

Table 5.5: Target and actual mesh sizes for C5G7 scaling test suite

Target Mesh Size	Actual Mesh Size	Relative Difference
262,144 ( $2^{18}$ )	271746	3.66%
524,288 ( $2^{19}$ )	543492	3.66%
1,048,576 ( $2^{20}$ )	1086984	3.66%
2,097,152 ( $2^{21}$ )	2173968	3.66%
4,194,304 ( $2^{22}$ )	4347936	3.66%
8,388,608 ( $2^{23}$ )	8695872	3.66%

Unlike the Godiva meshes, the C5G7 meshes are all the same percentage different from their respective target size. This is a consequence of a feature of this problem’s configuration that was implemented so that increasing mesh size of C5G7 would be from increased problem volume rather than mesh refinement, while still representing the same physical problem. To achieve this, we create a geometry that is uniform in the  $z$ -direction, the  $z$ -direction being the dimension that is parallel to the axes of the fuel rods. Then, we apply reflective boundary conditions to both  $z$ -axis planes, making our 3-D model physically equivalent to the 2-D C5G7 configuration, as the geometry is infinite and uniform along the  $z$ -axis. We then mesh a thin, one tetrahedron-thick layer of the geometry. To increase the mesh size, we then simply use the layered extrusion capability of GMSH, stacking layers on top of each other to increase the mesh size, while the physical model remains the same due to the reflective boundary conditions on the  $z$ -axis. The intended consequence of this procedure is that, as the mesh size increases, cells will not become smaller, nor will the modeled problem change. Thus, increasing mesh size with C5G7 does not result in the increase in required iterations that was observed with Godiva. Therefore, with these two problems used for scaling tests, we are able to observe the scaling trends of *PBJ-ITMM* and *IPBJ* in THOR both with (Godiva) and without (C5G7) the iterative slowdown that occurs with optical thinning of the cells due to mesh refinement. Three sample meshes used for the C5G7 tests are presented in Figs. 5.29 - 5.31, with the material legend for these meshes provided in Fig. 5.32.

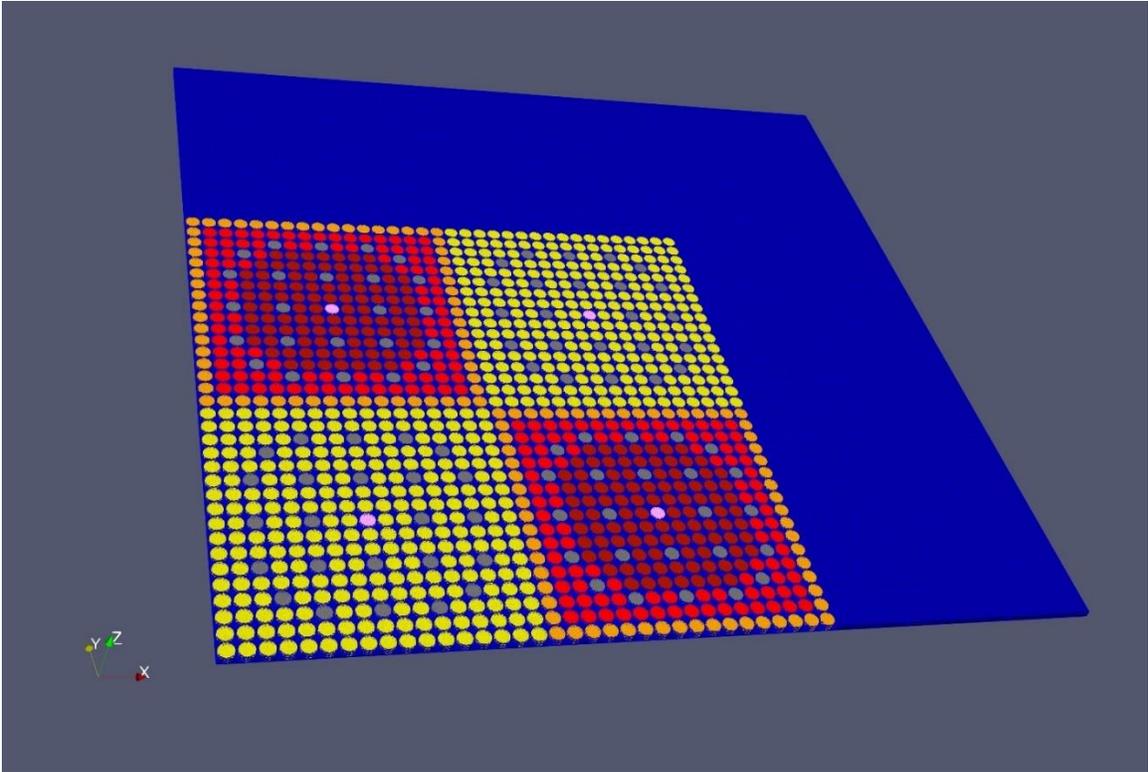


Figure 5.29: C5G7 Benchmark problem's mesh with  $\sim 524,288$  tetrahedral cells

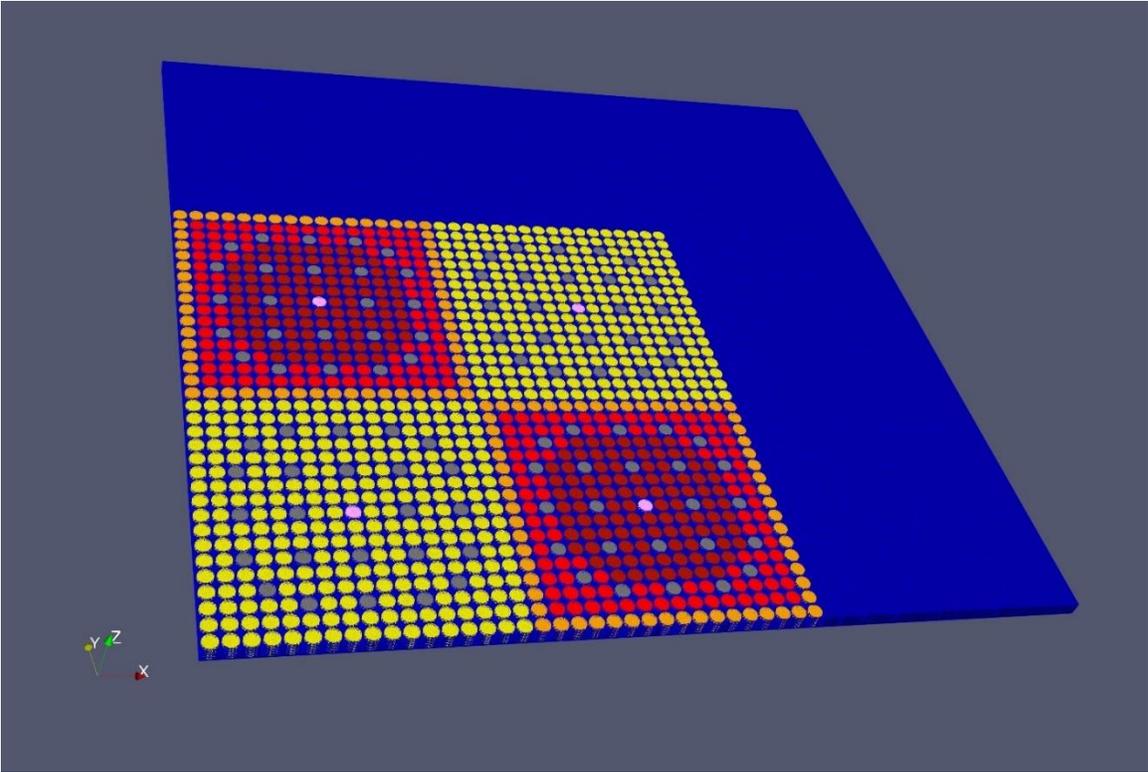


Figure 5.30: C5G7 Benchmark problem's mesh with  $\sim 1,048,576$  tetrahedral cells

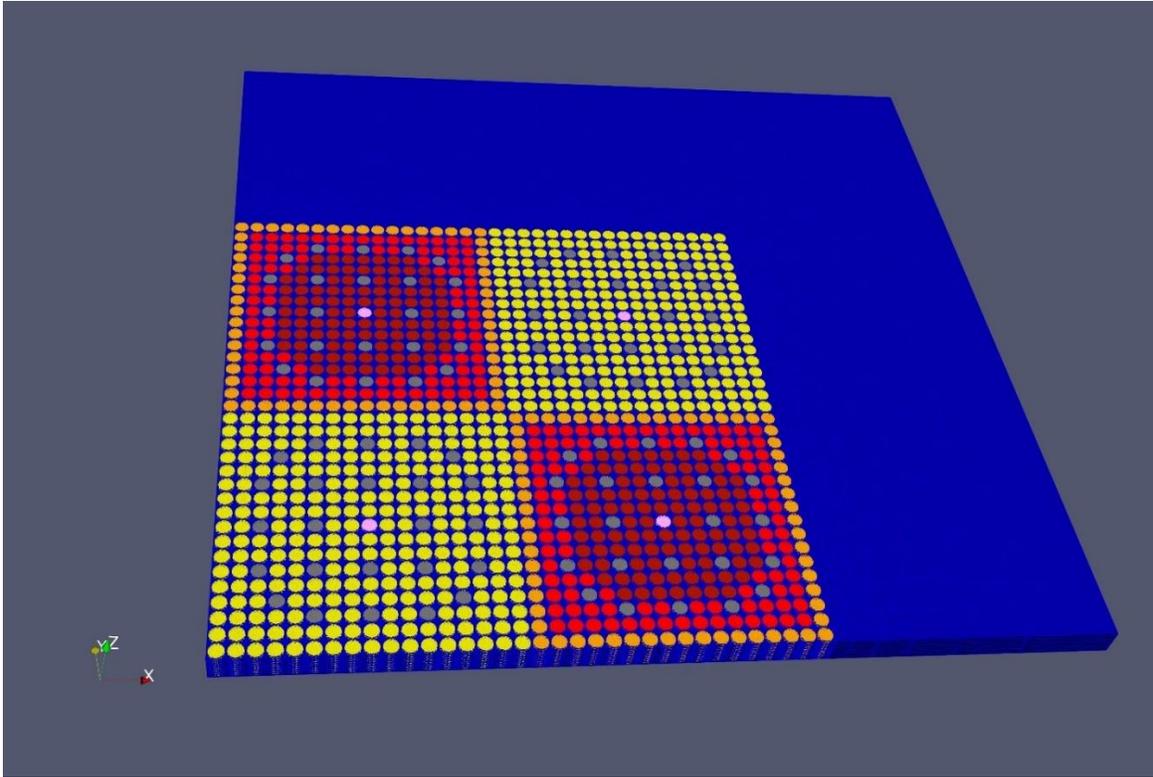


Figure 5.31: C5G7 Benchmark problem's mesh with ~2,097,152 tetrahedral cells

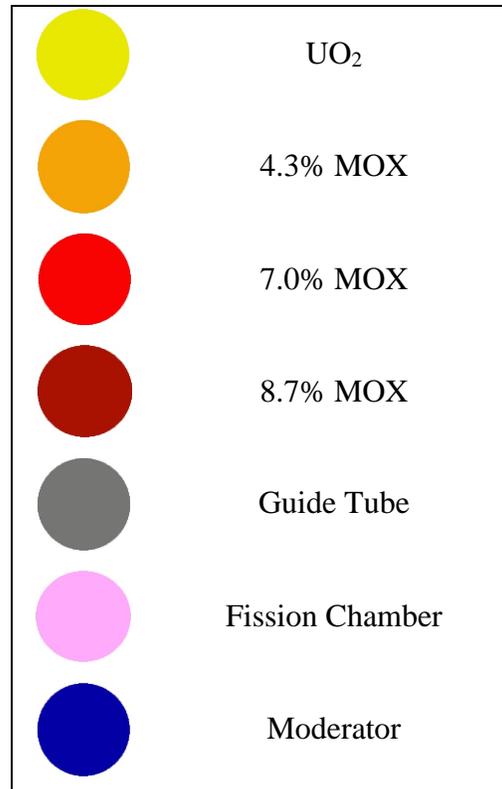


Figure 5.32: C5G7 mesh material legend

The three meshes presented vary only in the number of layers present. This is visible with each mesh being of a geometry twice as thick as the previous mesh's. The increasing problem volume with increasing mesh size as well as a visualization of the discretized problem configuration are the main features visible in these figures. Due to the large number of mesh cells, the actual meshing of the surface is difficult to observe. We therefore provide two additional images, Figs. 5.33 and 5.34, of the meshing of various type pins that are repeated in the C5G7 geometry. Additionally, we present three different partitioned meshes of C5G7 in Figs. 5.35 - 5.37. Note that in the visualization of these partitions, sub-domains in the surrounding moderator appear to vary drastically in the number of constituent cells. This is only a visual effect, with these sub-domains frequently continuing into the assembly region, where the color appears brighter due to the decreased cell size.

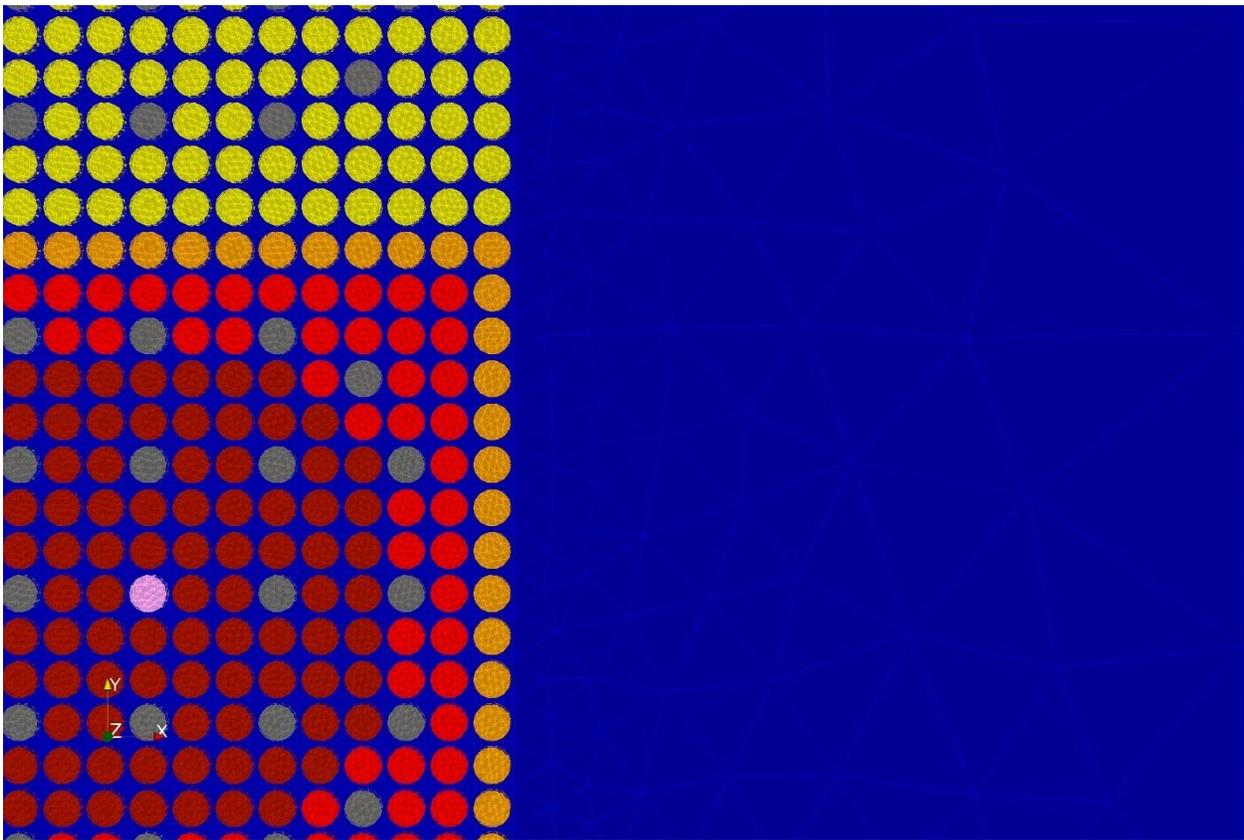


Figure 5.33: C5G7 Benchmark problem's tetrahedral mesh at fuel assembly, moderator interface

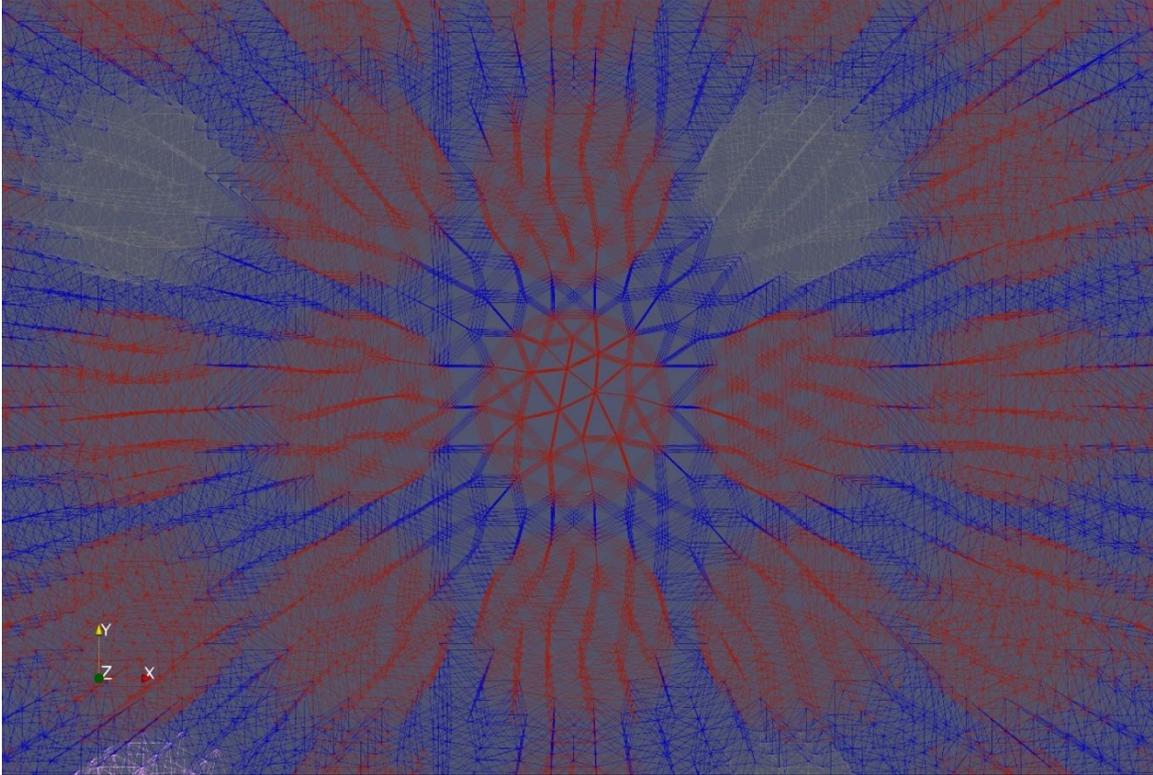


Figure 5.34: Tetrahedral mesh of a C5G7 Benchmark problem's fuel cell

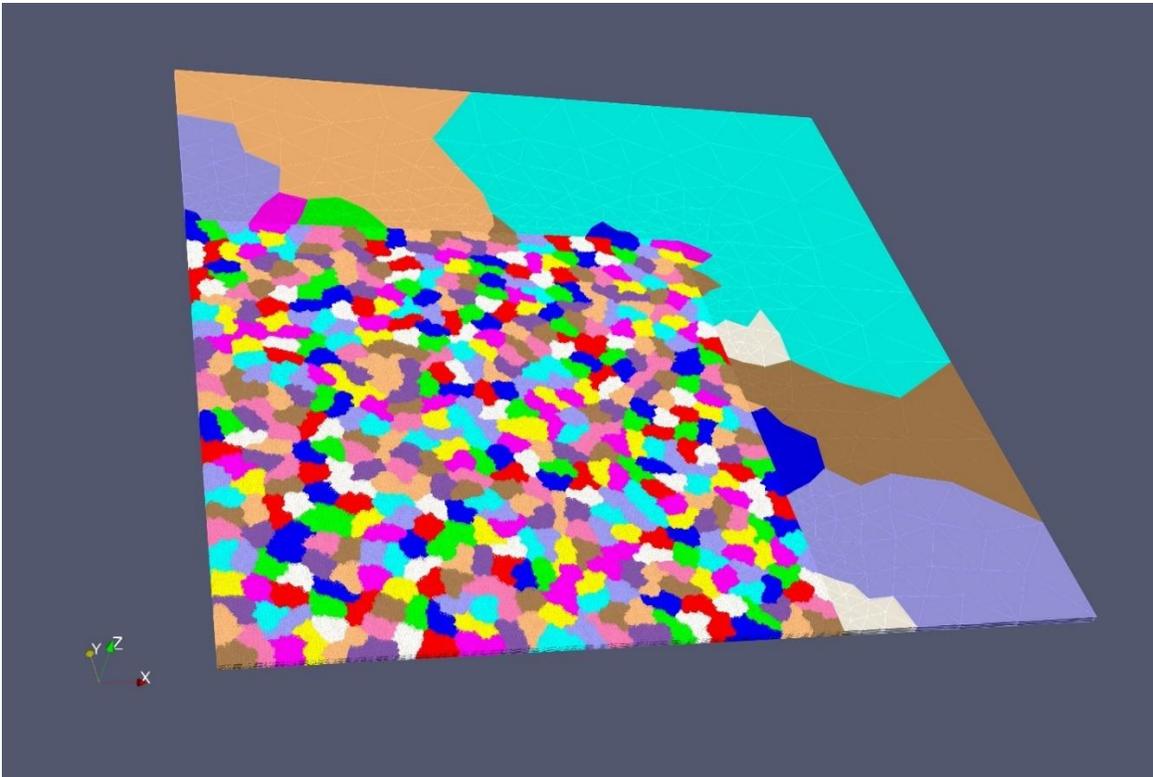


Figure 5.35: C5G7 partitioned mesh, ~524,288 cells, 512 sub-domains

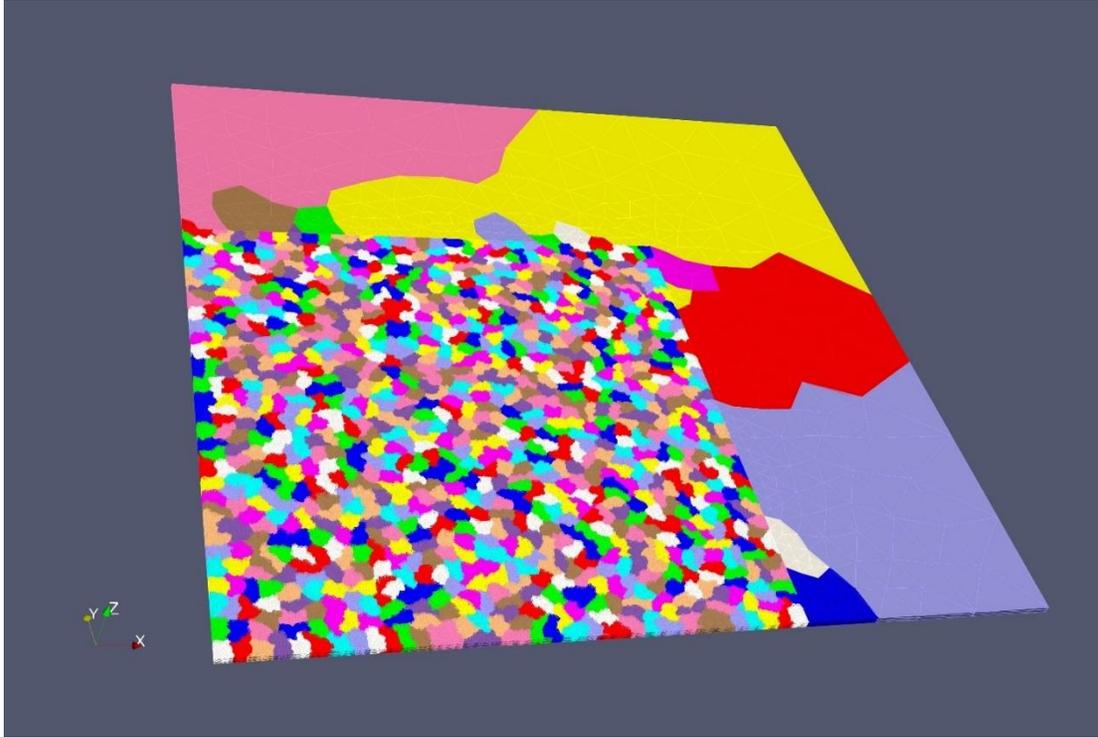


Figure 5.36: C5G7 partitioned mesh, ~524,288 cells, 1024 sub-domains

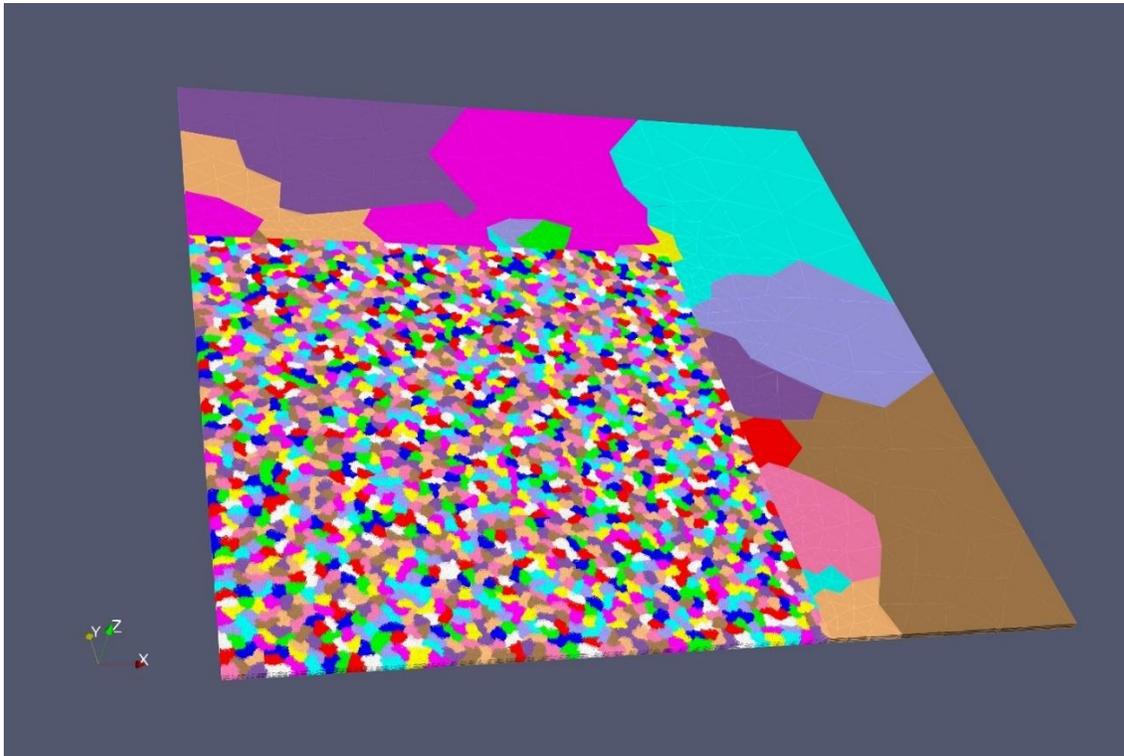


Figure 5.37: C5G7 partitioned mesh, ~524,288 cells, 2048 sub-domains

With the meshes generated and partitioned for each case in the C5G7 test suite, all are executed on Sawtooth. As with the Godiva test suite, we precede performance results with examining the  $k$ -eigenvalues produced by THOR for C5G7 as a function of mesh size for verification purposes, Fig. 5.38. The C5G7 THOR inputs are verified by this plot, with a computed eigenvalue (1.19254) that is within the range provided in the 2-D versions of this benchmark's literature. [72] Additionally, the fact that the eigenvalue does not change with mesh size indicates that our procedure of adding layers to increase mesh size creates physically identical problem configurations. Finally, with the produced eigenvalues for a given mesh size not varying with the method or number of partitions used, as well as comparison of this eigenvalue to that produced by the previously developed serial version of THOR, we judge our *PBJ* implementation to be verified.

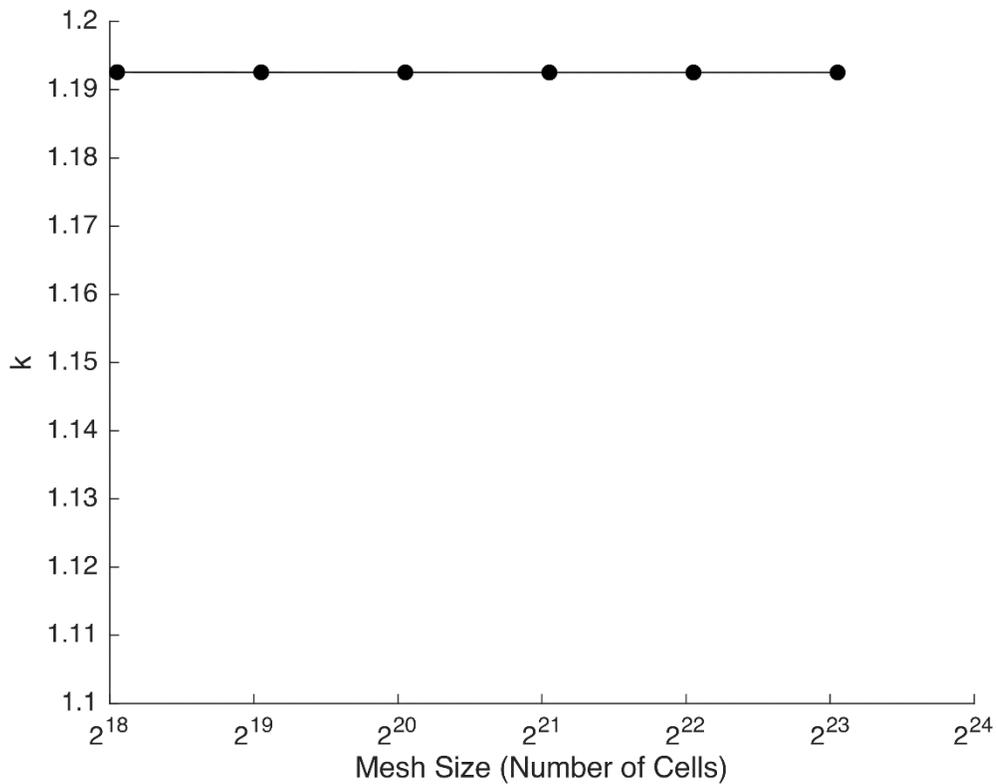


Figure 5.38:  $k$ -eigenvalues for C5G7 produced by THOR vs. mesh size

The presentation order of performance results follows that of Godiva, with the first timing metric we consider being *GFIC* construction time for various sub-domain sizes. The construction times for each C5G7 case are displayed in Fig. 5.39. Note that each point color in this plot

represents a different global mesh size. We omit a legend from this plot, however, since the global mesh size should not affect the construction time, as this is a local process. We include the data points from all mesh sizes to demonstrate that the observed construction time follows this expectation and is only dependent on sub-domain size.

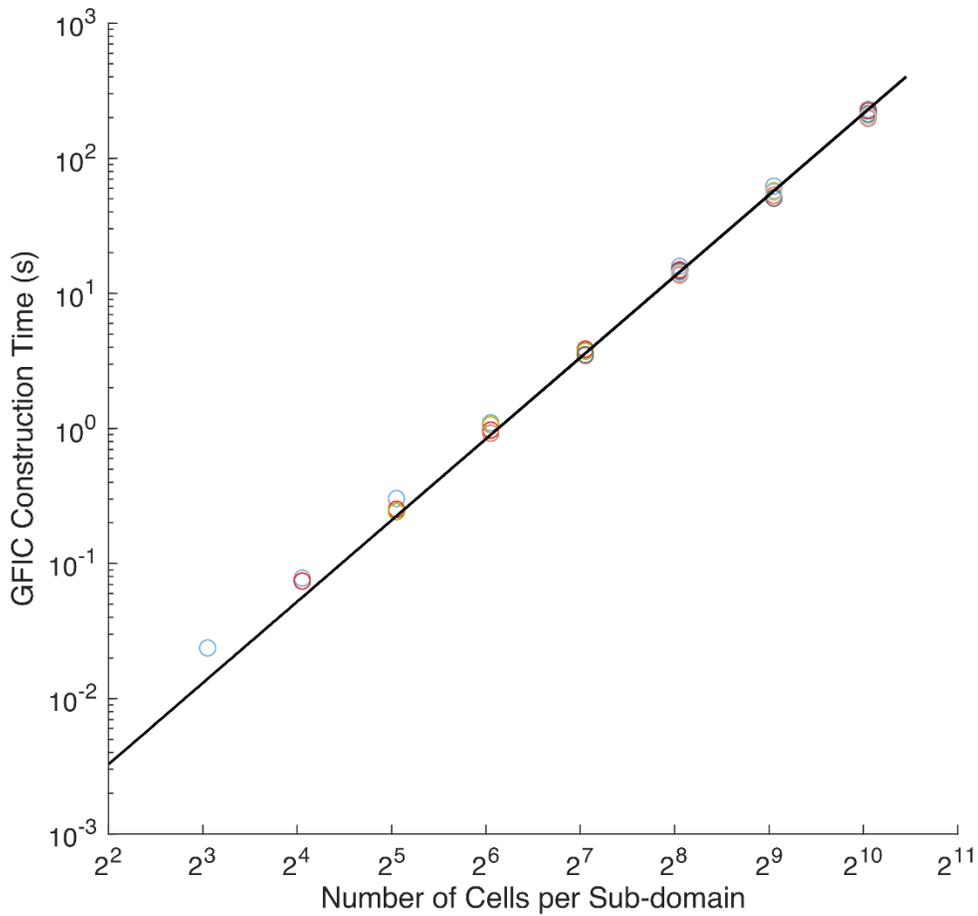


Figure 5.39: Time consumed by *GFIC* to construct *ITMM* matrices for C5G7 in THOR (points) and leading order of theoretical construction time, Eq. (3.88) (line)

The measured *GFIC* construction cost trend in this figure is consistent both with theoretical prediction (Eq. (3.88)) and the measured trend from Godiva (Fig. 5.12). The only observable difference between the construction cost for C5G7 and Godiva is that C5G7's is universally higher due to the extra energy group compared to Godiva. Otherwise, the conclusions are identical to those of Godiva's results, further verifying the predicted trend as well as further establishing the

effectiveness of this approach, with sub-domains 256 cells or less all construction is under one minute. The next set of results for our C5G7 scaling tests is the strong scaling results, presented in Figs. 5.40 - 5.47. These strong scaling results are the first four rows of Table 5.4, accounting for meshes up to ~2,097,152 cells in size. Full iteration and timing data from the C5G7 testing suite, Table 5.4, is available in Appendix D.

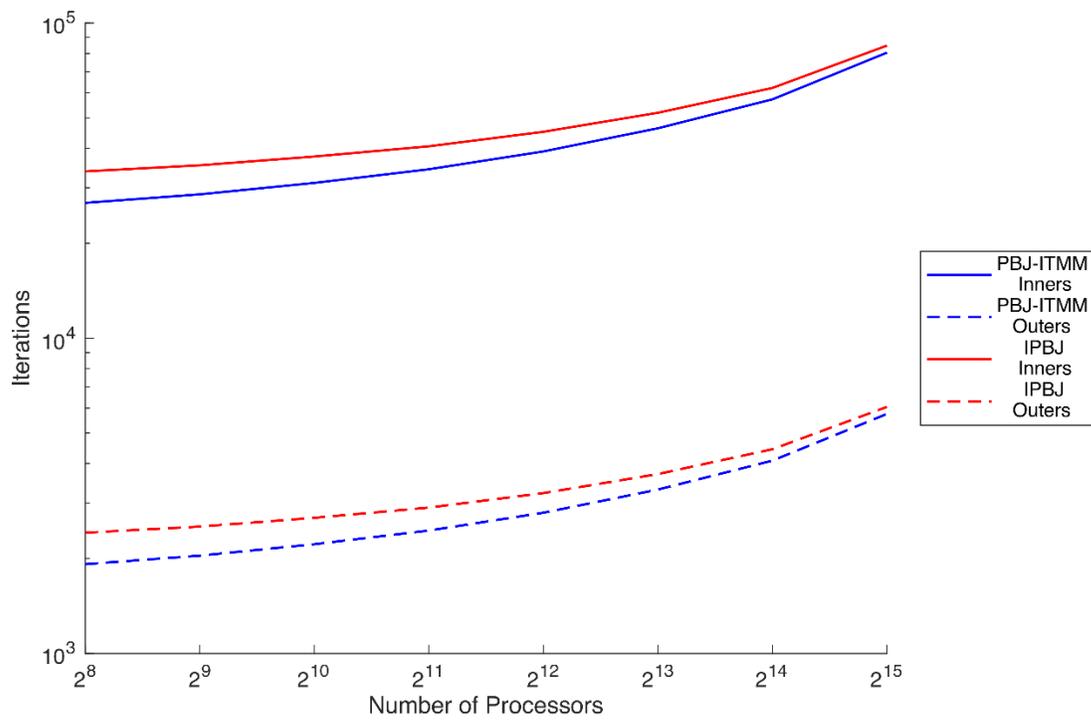


Figure 5.40: Iteration strong scaling for C5G7: ~262,144 cell mesh

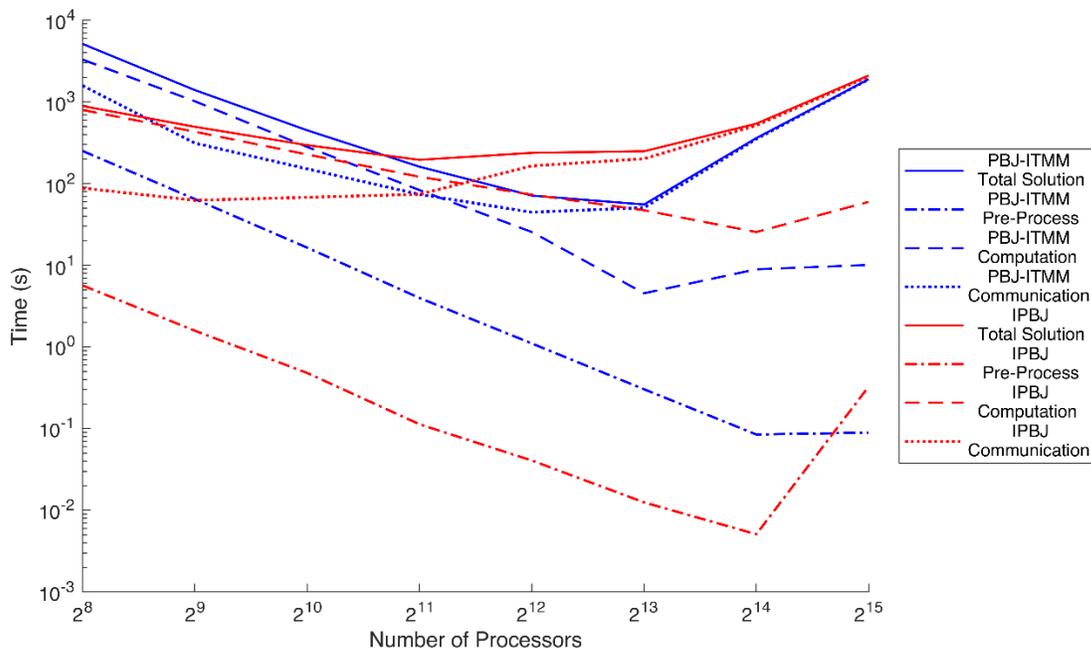


Figure 5.41: Solution time strong scaling for C5G7: ~262,144 cell mesh

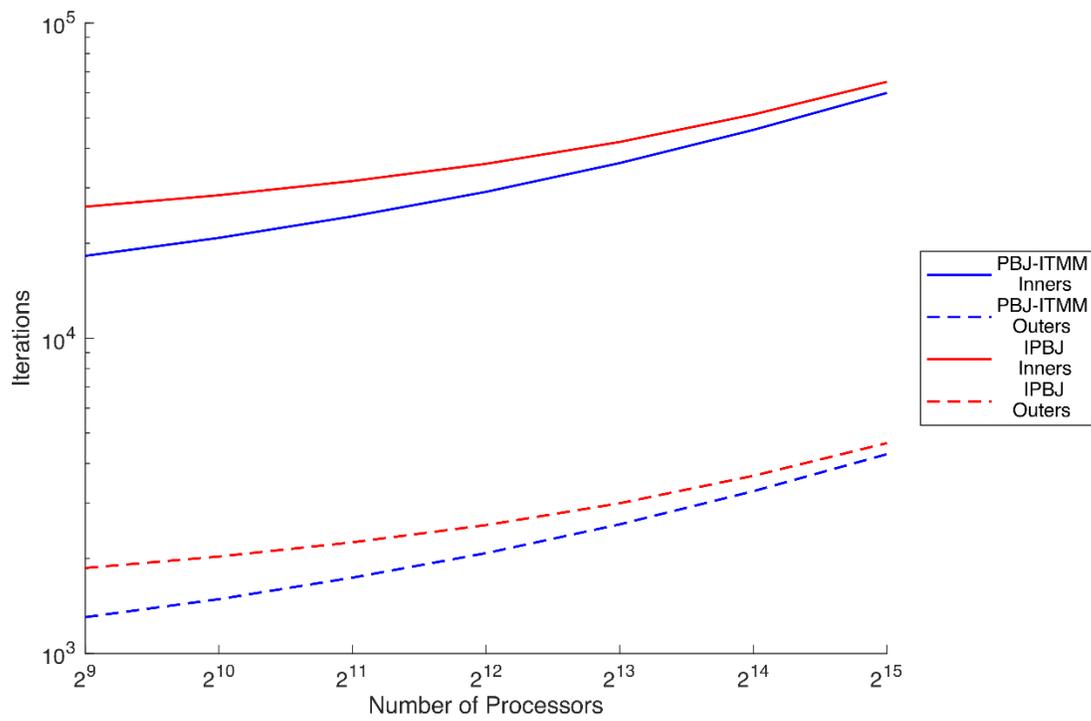


Figure 5.42: Iteration strong scaling for C5G7: ~524,288 cell mesh

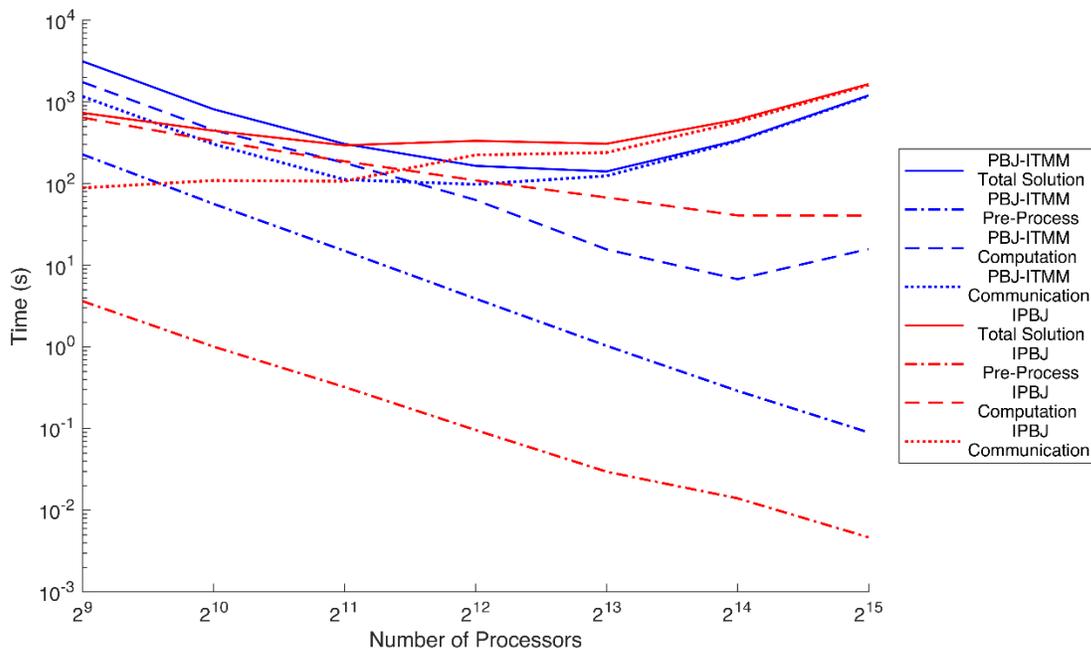


Figure 5.43: Solution time strong scaling for C5G7: ~524,288 cell mesh

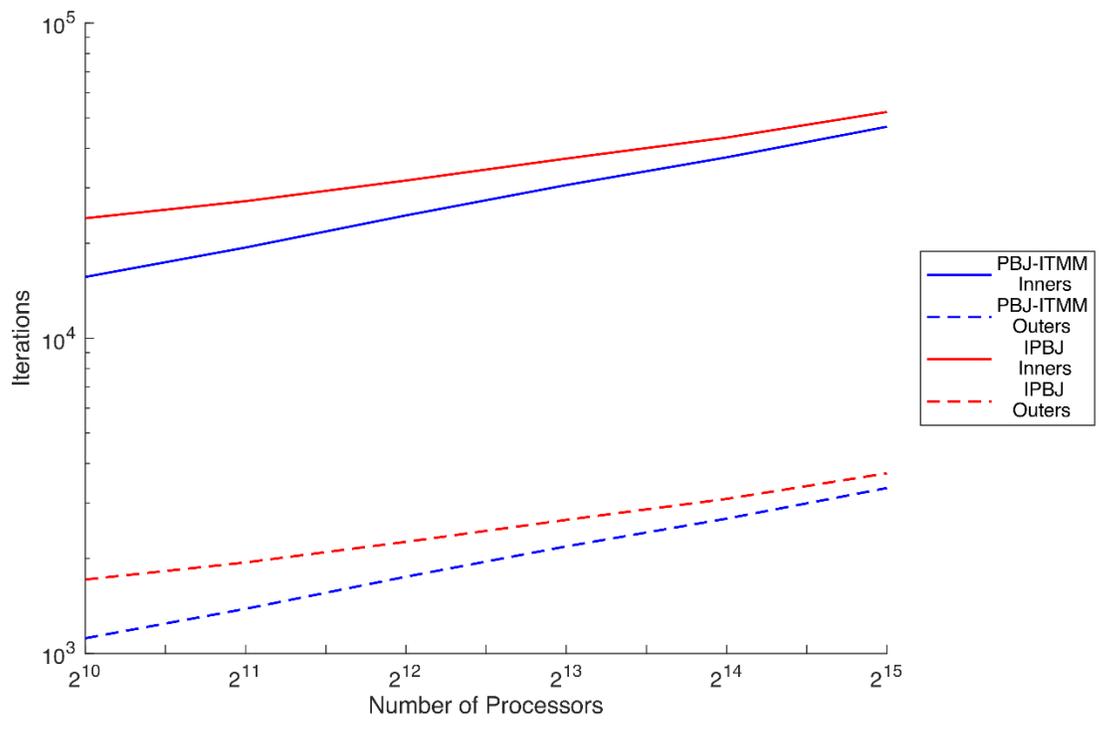


Figure 5.44: Iteration strong scaling for C5G7: ~1,048,576 cell mesh

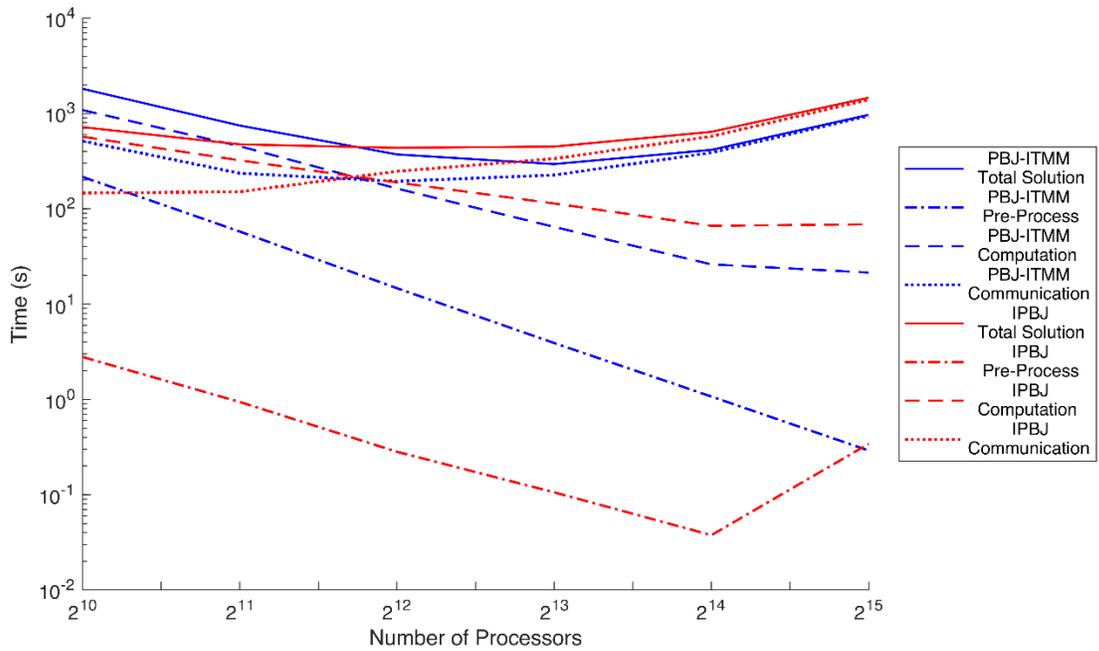


Figure 5.45: Solution time strong scaling for C5G7: ~1,048,576 cell mesh

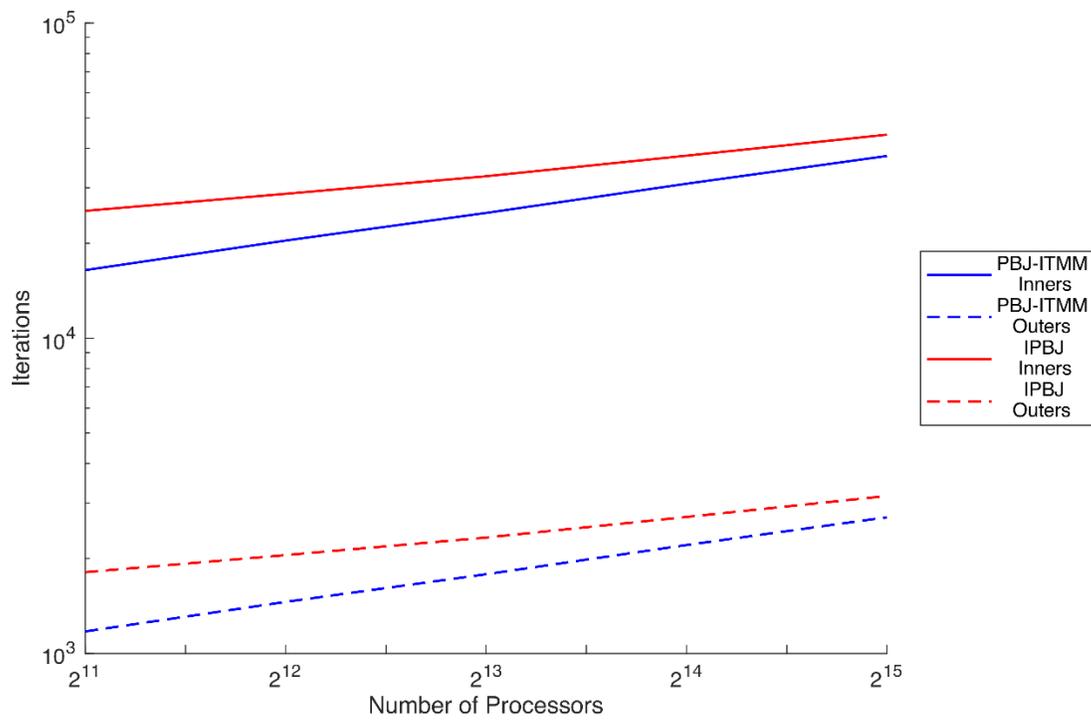


Figure 5.46: Iteration strong scaling for C5G7: ~2,097,152 cell mesh

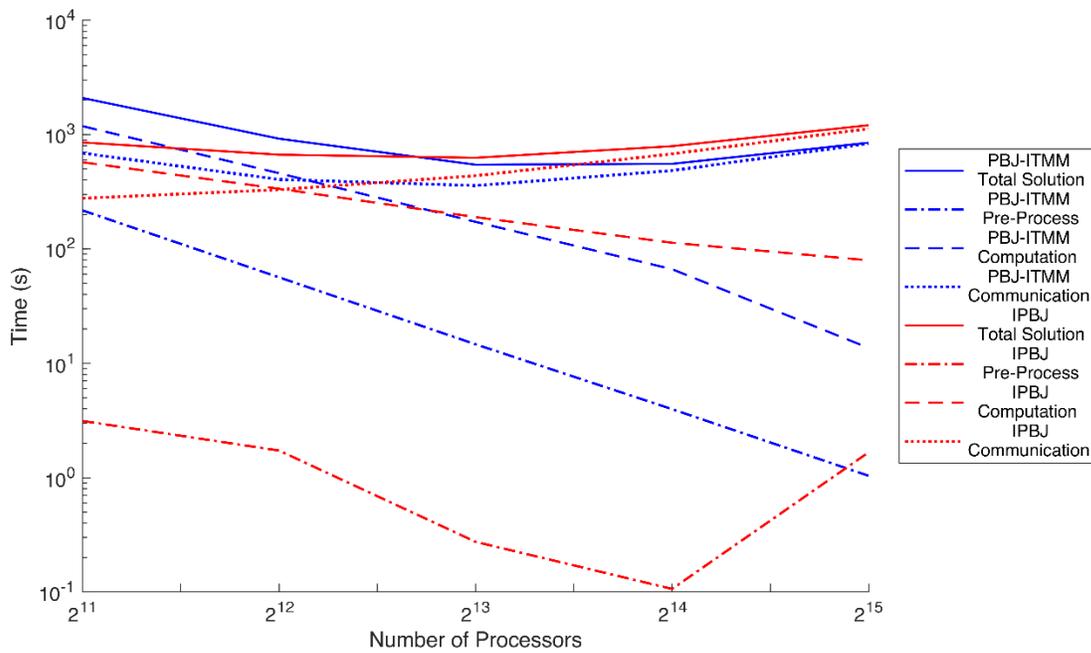


Figure 5.47: Solution time strong scaling for C5G7: ~2,097,152 cell mesh

The C5G7 strong scaling trends generally yield similar conclusions to those of the Godiva tests, with an increased advantage for *PBJ-ITMM* compared to *IPBJ*. Considering the strong scaling iteration trends, Figs. 5.40, 5.42, 5.44, and 5.46, *PBJ-ITMM*, as expected, universally requires fewer iterations to converge than *IPBJ*, with the relative difference decreasing as processors are added for a given mesh size. This decrease in the relative difference in required iterations between the two methods is due to the thinning sub-domains rendering a scenario in which the resolution of localized scattering is less significant compared to the asynchronicity. The same trend was observed with the Godiva strong scaling tests, and it indicates that the utility of *PBJ*-type methods for transport solution lies in the scalability to meshes of large size. Compared to Godiva, C5G7 requires far more iterations for most cases, with a higher degree of coupling between energy groups due to the cycle of moderation from fast groups to thermal, and fission from thermal groups to fission.

Continuing to the analysis of the solution time strong scaling trends, Figs. 5.41, 5.43, 5.45, and 5.47, the increase in required iterations compared to Godiva results in the pre-process time for *PBJ-ITMM* being universally the smallest contribution to total solution time. In fact, the pre-process time is comparatively small enough that it is largely insignificant. The construction time required by *GFIC* is independent of the number of subsequently required iterations, and is therefore less significant compared to the iterative solution time when the number of iterations is large. The remaining two contributions to solution time, computation and communication, are the largest contributors at low and high processor counts, respectively. The degree to which communication dominates the total solution time increases with increased mesh size. All of these general trends extend to *IPBJ* as well, but with the added observation that the pre-process time for *IPBJ* is small enough to be negligible across all cases.

Comparing the time components strong scaling trends of *PBJ-ITMM* and *IPBJ*, the general observations are consistent with those of the Godiva tests. At low processor counts, the super-linear scaling of *PBJ-ITMM*'s iterative solution time with respect to sub-domain size results in *IPBJ* converging faster, with the difference in required iterations being overwhelmed by *PBJ-ITMM*'s increased grind time. As processor count increases, however, *PBJ-ITMM*'s grind time decreases and communication time becomes dominant, both effects resulting in *PBJ-ITMM*'s performance improving relative to *IPBJ*'s. With the increased number of iterations of C5G7 relative to Godiva, though two effects result in *PBJ-ITMM* becoming the faster method at a larger

sub-domain size than was previously observed with Godiva. With the increased number of iterations, the absolute difference between the number required by *PBJ-ITMM* and *IPBJ* increases. Additionally, the pre-process cost is insignificant, which was observed to be significant, and in some cases, dominant in the Godiva tests. The consequence of these effects is that sub-domain size, below which *PBJ-ITMM* is the faster method is above 256 cells for C5G7, compared to ~128 with Godiva.

With the fundamental features of the strong scaling C5G7 results being consistent with those of Godiva and with the differences predictable and reasonable, we proceed to the weak scaling tests. The weak scaling trends are generated for sub-domain sizes ranging from ~64 to ~256 cells from the results of the associated columns of Table 5.4. This range constitutes the sub-domain sizes with which *PBJ-ITMM* is preferable to *IPBJ* and where there are also enough data points to compose a meaningful trend. These weak scaling trends are presented in Figs. 5.48 - 5.53.

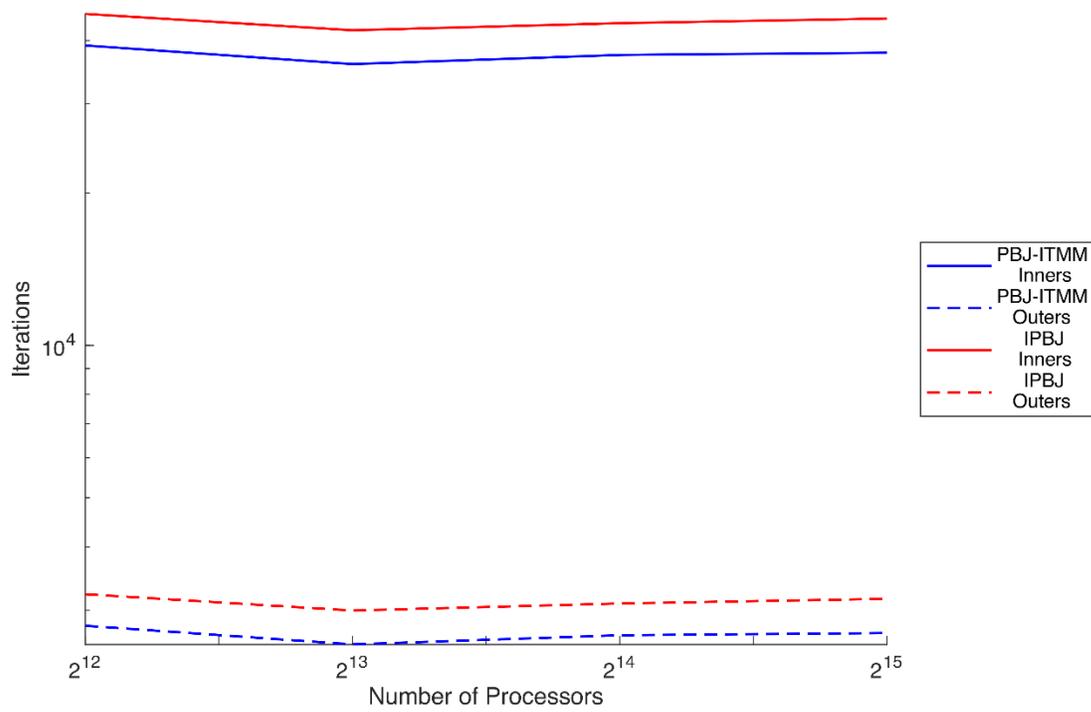


Figure 5.48: Iteration weak scaling for C5G7: ~64 cells per sub-domain (processor)

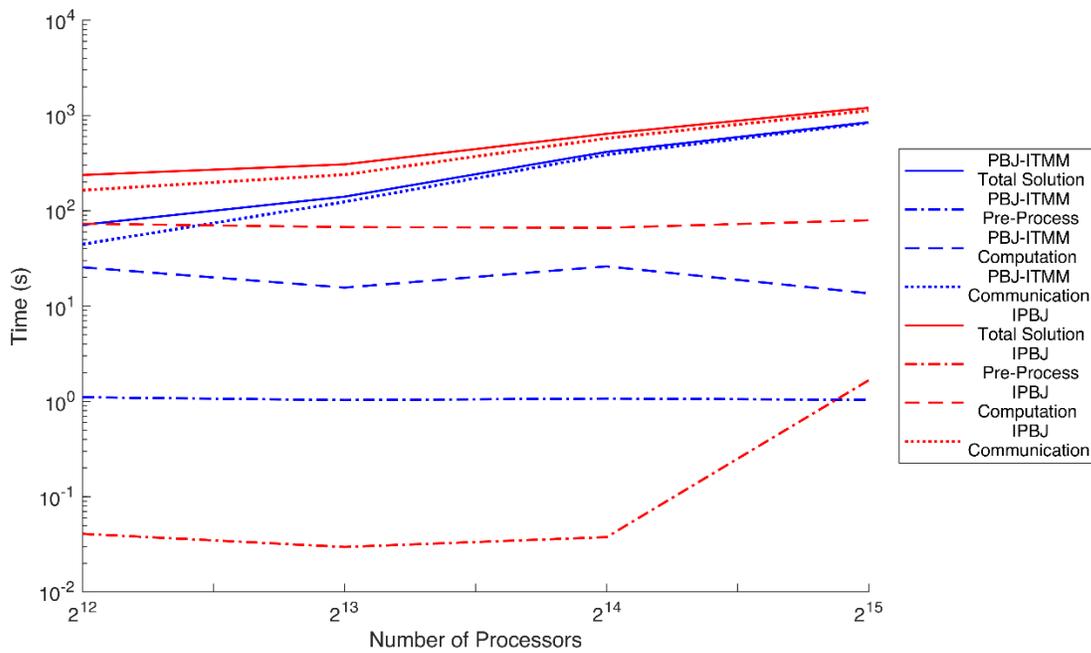


Figure 5.49: Solution time weak scaling for C5G7: ~64 cells per sub-domain (processor)

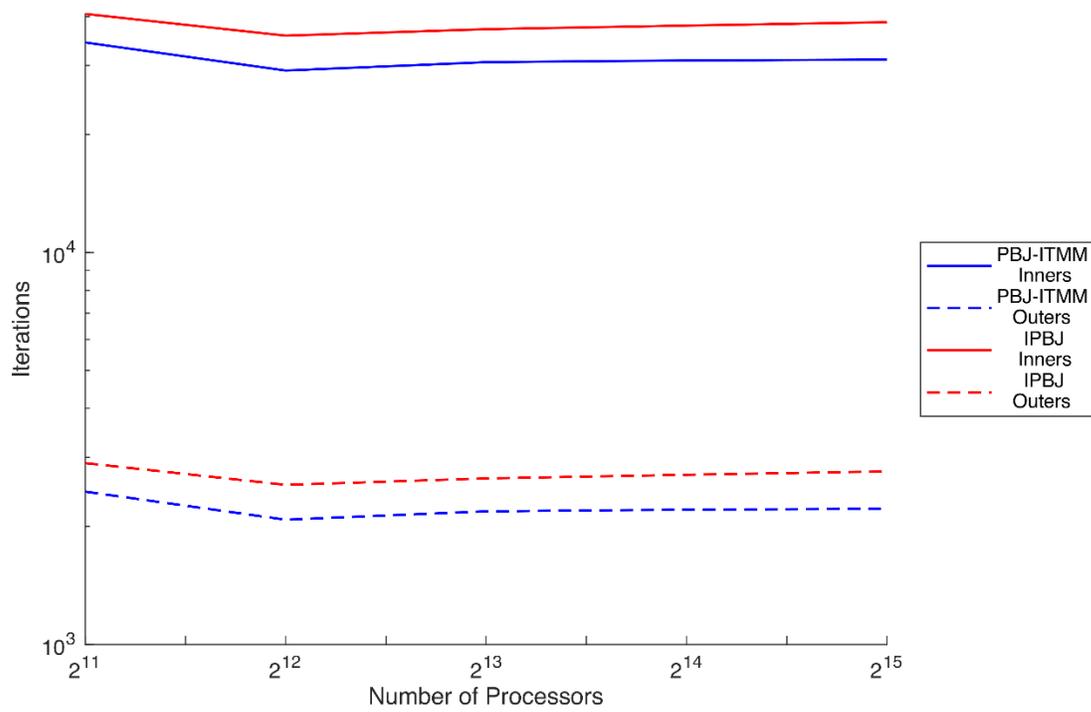


Figure 5.50: Iteration weak scaling for C5G7: ~128 cells per sub-domain (processor)

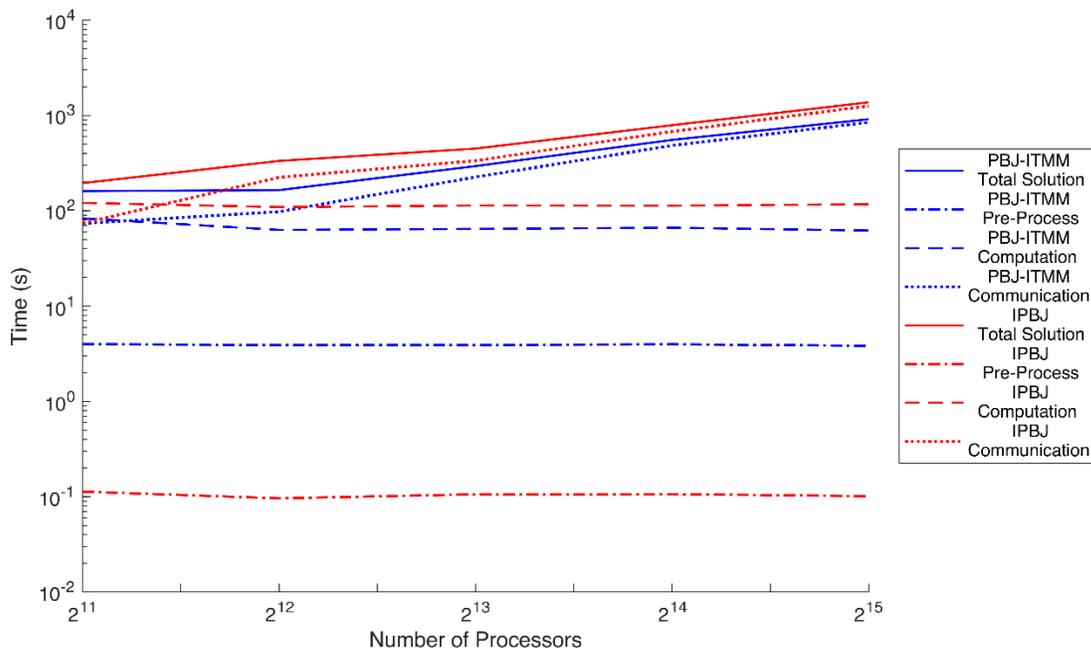


Figure 5.51: Solution time weak scaling for C5G7: ~128 cells per sub-domain (processor)

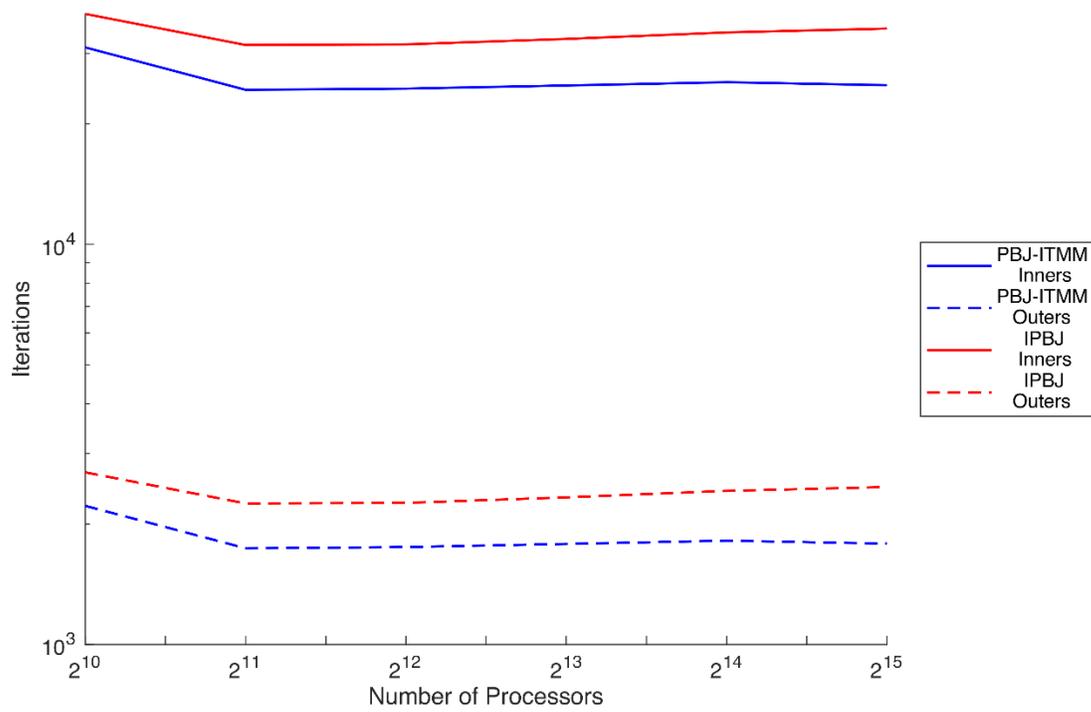


Figure 5.52: Iteration weak scaling for C5G7: ~256 cells per sub-domain (processor)

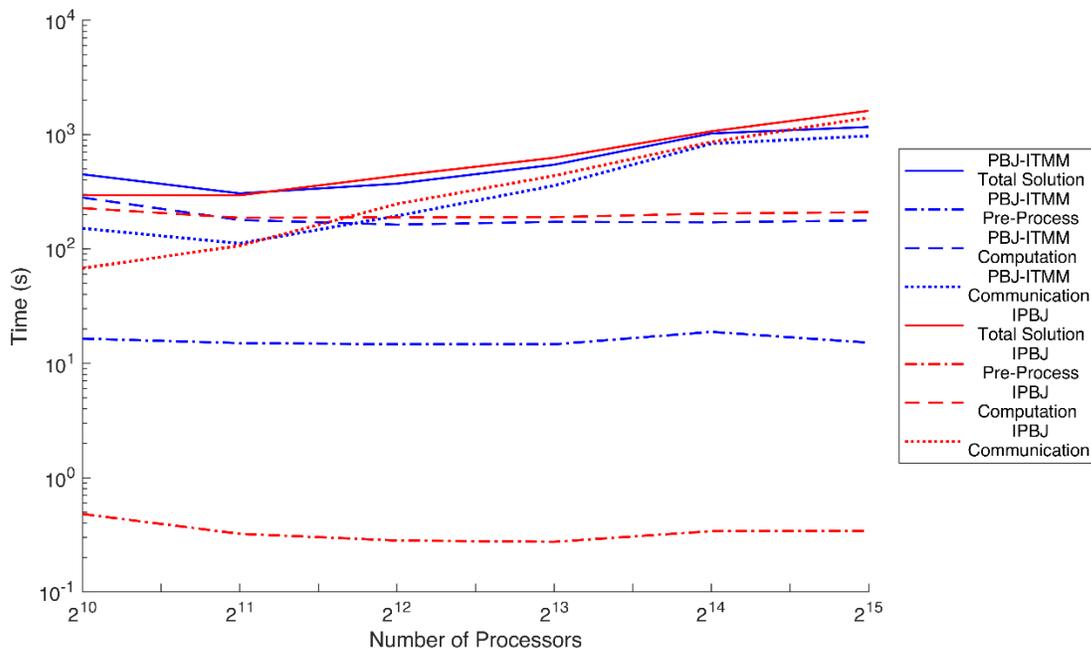


Figure 5.53: Solution time weak scaling for C5G7: ~256 cells per sub-domain (processor)

Contrasting the C5G7 and Godiva weak scaling results, the difference in the impact of *PBJ*-type methods' iterative performance between increased mesh size due to refinement and the addition of layers to increase the problem's volume is visible. The C5G7 weak scaling iterative trends, Figs. 5.48, 5.50, and 5.52, are mostly constant with respect to processor count for a given number of cells per sub-domain. This is a consequence of the fact that the physical volume increases proportionately to the number of cells in the mesh, rendering the optical thickness of sub-domains mostly constant for all cases on a single weak scaling trend. The only significant effect on the number of required iterations from the processor count for the C5G7 weak scaling trends is that there is actually an initial decrease in iterations as the processor count first increases. This is simply due to increased regularity of sub-domain shapes as layers are added to the problem. When the number of layers is small, sub-domains are, on average, skinnier in the  $z$ -direction, as this dimension is only a few cells thick.

The iteration trends for the C5G7 weak scaling tests demonstrate that the prescribed increasing of the geometric volume by adding layers accomplished the intended objective of keeping the number of required iterations relatively constant for a given number of cells per sub-domain. Consequently, when we analyze the associated solution time weak scaling trends, Figs. 5.49, 5.51, and 5.53, the increasing solution time with increased processor count should be attributable only to the non-ideal scaling of communication cost (ideal scaling being constant with respect to processor count). This is clearly the case, with pre-process time largely constant with respect to processor count, and with computation time almost constant, but with an initial decrease following the iteration trend, for both *PBJ-ITMM* and *IPBJ*. Communication, on the other hand, generally increases with increased processor count with the exception of the range over which the initial decrease in iterations occurs. As the communication time increases, some of the cases observed begin to increase at a linear rate on the  $\text{Log}_{10}\text{-Log}_2$  scale for several data points, suggesting these cases have entered an asymptotic regime in which the increase in communication cost associated with an increased processor count is predictable. For both *PBJ-ITMM* and *IPBJ*, once this asymptotic regime is entered, in order for the communication cost to increase by roughly an order of magnitude, the processor count must increase by a factor of eight. With this observation, if we consider the weak scaling trends that extend to the maximum processor count available, 32,768, if this trend were to continue, both methods would extend well into the massively parallel regime to 262,144 processors with execution times under three hours on

problems of this geometric configuration with mesh sizes of over 67 million cells. Verifying this prediction would require access to larger allocations of processors than were accessible during this work.

Comparing the two methods, the solution time weak scaling trends are consistent with the strong scaling trends on the relative performance of *PBJ-ITMM* and *IPBJ*. *PBJ-ITMM* executes almost universally faster than *IPBJ* over the given trends for the observed sub-domain sizes.

The weak and strong scaling trends for C5G7 support the conclusion from the Godiva tests that *PBJ*-type methods, due to their asynchronicity, are best fitted to solving very large problems, rather than excessively parallelizing a problem of modest size. Additionally, both testing suites demonstrate *PBJ-ITMM* to be faster than *IPBJ* when sub-domains are below a certain number of cells and vice-versa when above. This threshold is found to be problem dependent, with Godiva's occurring at ~128 cells per sub-domain and C5G7's occurring at more than 256 cells per sub-domain. The increased range over which *PBJ-ITMM* is preferable to *IPBJ* for C5G7 compared to Godiva is a positive indication for *PBJ-ITMM*'s performance. As discussed, this effect is largely due to C5G7 being a more difficult problem to converge, with increased mutual coupling between energy groups. This result therefore indicates that *PBJ-ITMM*'s favorability is greater in more difficult problems, greatly increasing its suggested utility.

The final data point of Fig. 5.53 represents the largest scale problem tested in this work, with the largest allocation of processors solving the largest mesh for the problem with the largest number of groups. This problem contains over 60 million cell-averaged scalar fluxes, over 1.4 billion cell/angle/group combinations, and over 6.8 billion total unknowns when the method of short characteristics is used. This problem converges in under 30 minutes with *IPBJ* and under 20 minutes with *PBJ-ITMM*, executing on 32,768 processors. This combined with the observed rate of increase in communication cost with respect to processor count suggest that our implementation is scalable to massively parallel solution. Finally, our new *GFIC* algorithm, which allowed the solution of *PBJ-ITMM* on unstructured grids is observed to be an effective response-matrices construction algorithm, with the construction time being insignificant for iteratively intensive problems such as C5G7.

## 5.7: CPU Computation Time Comparison to Synchronously Sweeping

The application of *PBJ*-type methods for transport calculations is largely sought as an alternative to *SI* for spatially parallel execution on unstructured grids. The asynchronicity of *PBJ* is warranted for such an application, as it comes with the benefit of avoiding the scheduling complications associated with *SI*, discussed in the literature review in Sec. 2.2.4.

Through the development and implementation of *GFIC* in this work, we have achieved parallel solution of *PBJ-ITMM* on unstructured grids and demonstrated its efficacy through scaling tests. With an alternative to spatially parallel *SI* on unstructured grids, this section presents a preliminary comparison between *PBJ-ITMM*, *IPBJ*, and *SI* on unstructured grids in serial operation. This comparison is serial because THOR does not currently have spatially parallel *SI* capabilities. The serial execution time for *PBJ-ITMM* or *IPBJ* is calculated by subtracting the communication time (including initial communication of instructions) from the total solution time, then multiplying by the number of processors. These times are compared to the total solution time required for serial execution with *SI* for ~262,144 cell meshes, along with the required number of iterations for all methods, for both Godiva and C5G7 in Tables 5.6 - 5.9.

Table 5.6: Number of required iterations to converge ~262,144 cell Godiva mesh

Cells per	1024	512	256	128	64	32	16	8	N/A
Sub-domain									
<i>PBJ-ITMM</i>	1140	1368	1668	2004	2448	3648	4668	5088	-
<i>IPBJ</i>	1392	1632	1920	2256	2700	3924	4980	5412	-
<i>SI</i>	-	-	-	-	-	-	-	-	384

Table 5.7: Serial CPU time (minutes) consumed to converge ~262,144 cell Godiva mesh

Cells per	1024	512	256	128	64	32	16	8	N/A
Sub-domain									
<i>PBJ-ITMM</i>	1029	605	351	222	111	109	363	166	-
<i>IPBJ</i>	142	160	191	222	282	421	627	696	-
<i>SI</i>	-	-	-	-	-	-	-	-	48

Table 5.8: Number of required iterations to converge ~262,144 cell C5G7 mesh

Cells per	1024	512	256	128	64	32	16	8	N/A
Sub-domain									
<i>PBJ-ITMM</i>	26,866	28,588	31,080	34,342	39,116	46,326	57,232	80,500	-
<i>IPBJ</i>	33,810	35,350	37,688	40,614	45,150	51,884	62,202	84,756	-
<i>SI</i>	-	-	-	-	-	-	-	-	23,478

Table 5.9: Serial CPU time (hours) consumed to converge ~262,144 cell C5G7 mesh

Cells per	1024	512	256	128	64	32	16	8	N/A
Sub-domain									
<i>PBJ-ITMM</i>	251	154	85	50	30	11	41	92	-
<i>IPBJ</i>	57	62	65	69	84	107	116	543	-
<i>SI</i>	-	-	-	-	-	-	-	-	47

These tables compare the required iterations and consumed CPU time neglecting communication (i.e. the time that would be required to execute in serial operation) for *PBJ-ITMM* and *IPBJ* to the required iterations and consumed solution time for serially executed *SI*. From Tables 5.6 and 5.8, *SI* converges Godiva and C5G7 in fewer iterations than both *PBJ-ITMM* and *IPBJ* for all sub-domain sizes tested. The magnitude of the difference in required iterations varies drastically with sub-domain size as well as which test problem is being solved. The relative difference between *SI* and *PBJ-ITMM* or *IPBJ* with respect to the number of required iterations is far greater for Godiva than C5G7, likely due to the larger amount of inter-group coupling due to increased production in C5G7 compared to Godiva. Additionally, as expected, the difference increases as sub-domain size decreases.

This increase in required iterations compared to *SI* notwithstanding, *PBJ-ITMM* still consumes CPU execution times that are competitive with *SI*'s, even solving C5G7 with less consumed CPU time when sub-domains are between 16 and 64 cells in size. *IPBJ* on the other hand consumes more CPU time than *SI* in all cases.

While this comparison suggests the faster method between *PBJ-ITMM* and *SI* to be problem and decomposition dependent, recall that this only a preliminary comparison, comparing the execution time in serial operation, not parallel. With *PBJ-ITMM* demonstrated to likely solve certain problems in shorter parallel execution times than *SI*, this comparison suggests the future

work of implementing both *PBJ-ITMM* and spatially parallel *SI* in the same unstructured grids code for performance comparison. Given the presented serial execution times, for a given case, the faster method will be determined by the addition of the execution time that is required for parallelization. For *PBJ-ITMM*, this is simply the addition of communication time, with the *GFIC* construction time included in the presented serial execution times. However, for *SI*, this is the addition of both communication time and the cost associated with the parallel scheduling algorithm. Additionally, the communication requirements for *PBJ-ITMM* and *SI* are different. Consider our 2-D Cartesian, four processor *KBA* example from the literature review, Fig. 2.1, the per-iteration *KBA* and *PBJ* communication requirements for which are displayed in Fig. 5.54, with dashed lines indicating sub-domain boundaries across which there is no between-processor communication and solid lines indicating sub-domain boundaries across which there is between-processor communication.

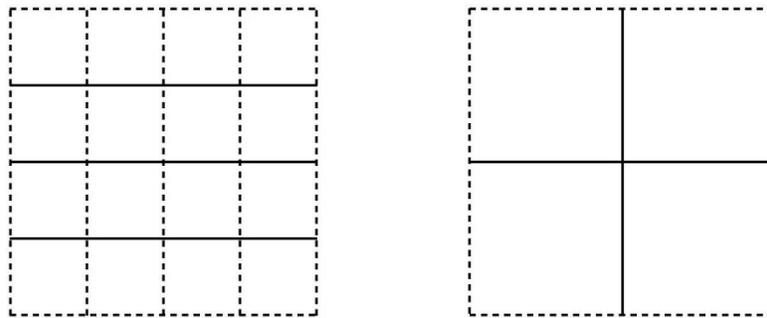


Figure 5.54: *KBA* (left) and *PBJ* (right) decompositions of a 2-D Cartesian spatial domain for solution on four processors with communicating (solid) and non-communicating (dashed) sub-domain boundaries

From this example, it is clear that while both decompositions presented are for execution on four processors, the amount of data per iteration that will be communicated between processors is 50% higher for *KBA* than for *PBJ*. This is a consequence of the fact that *PBJ* decomposes the domain into  $N_p$  sub-domains, while *KBA* decomposes the domain into  $N_p$  sub-domains along the diagonal, resulting in a larger number of smaller sub-domains than *PBJ*,  $N_p$  being number of processors. While *PBJ* can decompose into multiple sub-domains per processor, these sub-domains would be interior to those of our example and would not increase the amount of data per iteration that must be communicated. Additionally, this comparison holds in 3-D as the added dimension's sub-domain interfaces would all be communicating with *KBA*. On unstructured grids,

the communication patterns for an arbitrary problem become unpredictable. However, since the general feature of spatially parallel *SI* requiring decomposition into more sub-domains than *PBJ* extends to unstructured grids, we conjecture the consequential increase in the amount of data per iteration that must be communicated between processors for *SI* compared to *PBJ* to also extend to unstructured grids. While we expect the amount of data communicated per iteration to be higher with *SI* than *PBJ-ITMM*, since the communication patterns of these methods are different, as is the number of required iterations, there can be no prediction as to which method will consume more communication time in parallel execution on unstructured grids. While direct comparison of *PBJ-ITMM* and *SI* in parallel execution on unstructured grids is beyond the scope of this work, the demonstration of *PBJ-ITMM* to be scalable to the massively parallel regime with construction of the associated matrices via *GFIC* utilizing pre-existing code capabilities, this work provides the necessary mathematical tools for such a comparison study.

The serial execution times presented in this section suggest that the relative grind times of *PBJ-ITMM* and sweep-based methods (*IPBJ* and *SI*) are likely different in THOR's 3-D unstructured grid solutions than HAT-2C's 2-D Cartesian grid solutions. HAT-2C's parametric study in Sec. 4.6 indicates that *PBJ-ITMM*'s grind time is significantly higher than *SI* and *IPBJ*'s, with the scaling from iterations to execution time larger for *PBJ-ITMM*. This feature does not necessarily extend to unstructured grids, though, with *PBJ-ITMM*'s matrix-based iterative solution being agnostic to mesh type. Contrarily, the mesh sweep's per-cell/angle solution employs a different kernel calculation on unstructured grids than on Cartesian grids. In the case of short characteristics, cells are split into multiple canonical tetrahedra. [41]

To investigate, we present THOR's grind times for *PBJ-ITMM* and *IPBJ* (same as *SI*'s) in Table 5.10. Grind time is the execution time required to obtain the iterative solution to one element of phase space (i.e. cell/angle/group combination). To calculate grind time, we divide the computation time by the number of required iterations, angles, energy groups, and cells per sub-domain. This calculation is performed on the Godiva results, as there are more data points for this problem than for C5G7. Since grind time varies with sub-domain size for *PBJ-ITMM*, we average the grind times calculated for the cases of a single column of Table 5.2 to produce the grind time for the associated sub-domain size. Since *IPBJ*'s grind time does not depend on sub-domain size, we simply average over all cases in Table 5.2.

Table 5.10: THOR grind times ( $\mu s$ ) for *PBJ-ITMM* and *IPBJ* (same as *SI*)

Cells per	1024	512	256	128	64	32	16	8	N/A
Sub-Domain									
<i>PBJ-ITMM</i>	0.445	0.263	0.151	0.0870	0.0423	0.0218	0.0387	0.0560	-
<i>IPBJ</i>	-	-	-	-	-	-	-	-	0.132

Table 5.10 displays *PBJ-ITMM*'s grind time in THOR to be less than that of *IPBJ* for sub-domains of size 128 or smaller. The two methods' grind times are comparable for 256 cell sub-domains. This trend of grind times is consistent with the scaling trends presented in the previous section, where the sub-domain sizes at which *PBJ-ITMM*'s total solution time became smaller than *IPBJ*'s were found to be  $\sim 128$  and  $\sim 256$  for Godiva and C5G7, respectively. We observe here that this is the region of sub-domain size over which *PBJ-ITMM*'s grind time becomes shorter than *IPBJ*'s. The final feature of the grind times presented is that *PBJ-ITMM*'s actually increases with decreasing sub-domain size when the number of cells per sub-domains is below 32. This is likely a consequence of the overhead cost associated with matrix allocation and the call to LaPACK.

## 5.8: Iterative Performance on Problem Containing Void Region

This chapter has detailed the implementation of *PBJ-ITMM* via *GFIC* and *IPBJ* for parallel solution on unstructured grids in the THOR code. In previous chapters, we have detailed the development, analysis, and testing of hybrid methods for mitigating the iterative slowdown experienced by *PBJ*-type methods in problems containing optically thin cells. While implementation of these hybrid methods for parallel solution on unstructured grids is beyond the scope of this work, we test *PBJ-ITMM* and *IPBJ* with THOR on a problem containing a void region to demonstrate that the observed iterative slowdown extends to unstructured grids, thus motivating the future work of implementing *AH-PI-IP* in THOR. The test problem used is the dog-leg void benchmark problem [67], the HAT-2C results for which can be found in Sec. 4.8. This test problem was meshed into  $\sim 420,000$  cells and partitioned into 4096 sub-domains. The mesh and partitioning are presented in Figs. 5.55 - 5.60. For the meshes, red shading indicates the void duct region containing a source, white shading indicates the source-free void duct region, and blue shading

indicates the surrounding shield material. All boundary conditions on external planes that bound the source region are reflective, all others are vacuum.

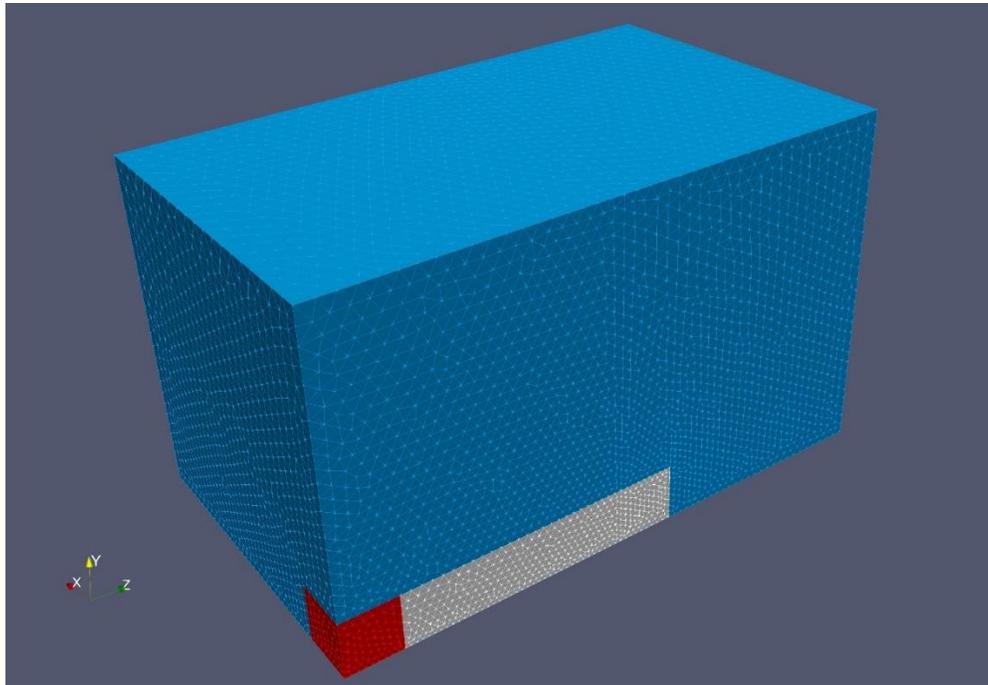


Figure 5.55: Dog-leg void problem mesh

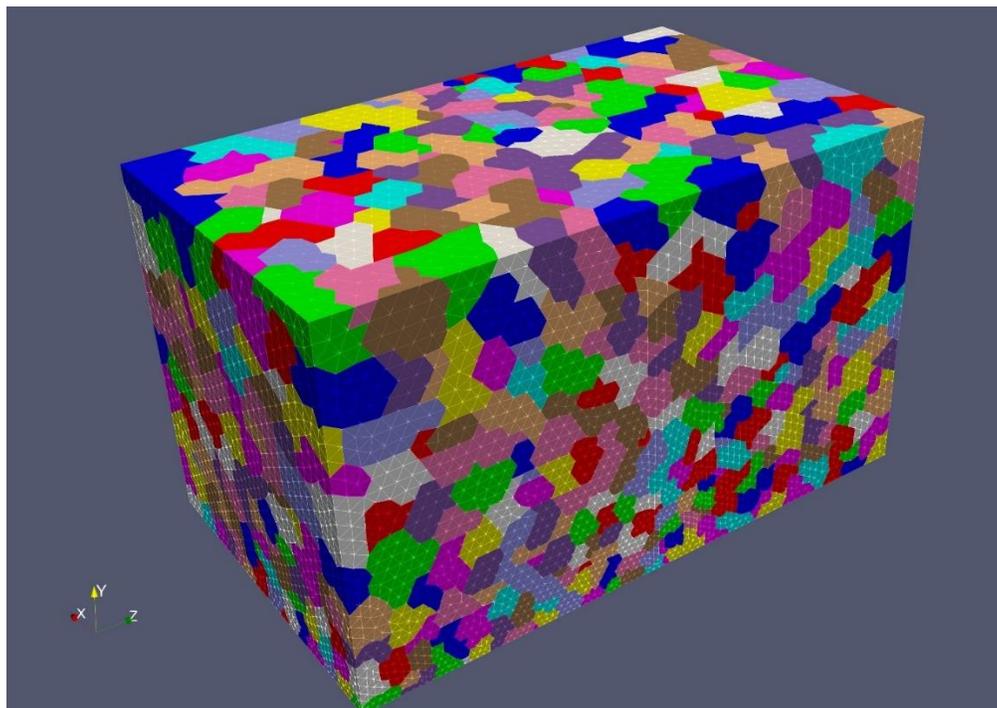


Figure 5.56: Dog-leg void problem partitioned mesh

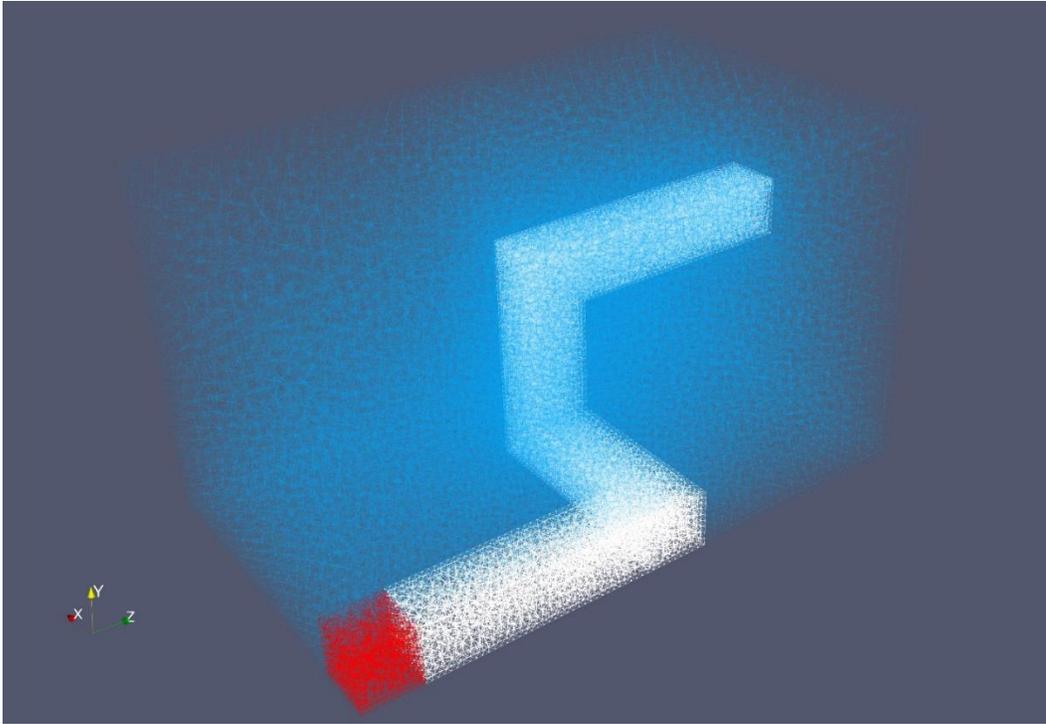


Figure 5.57: Dog-leg void problem interior mesh

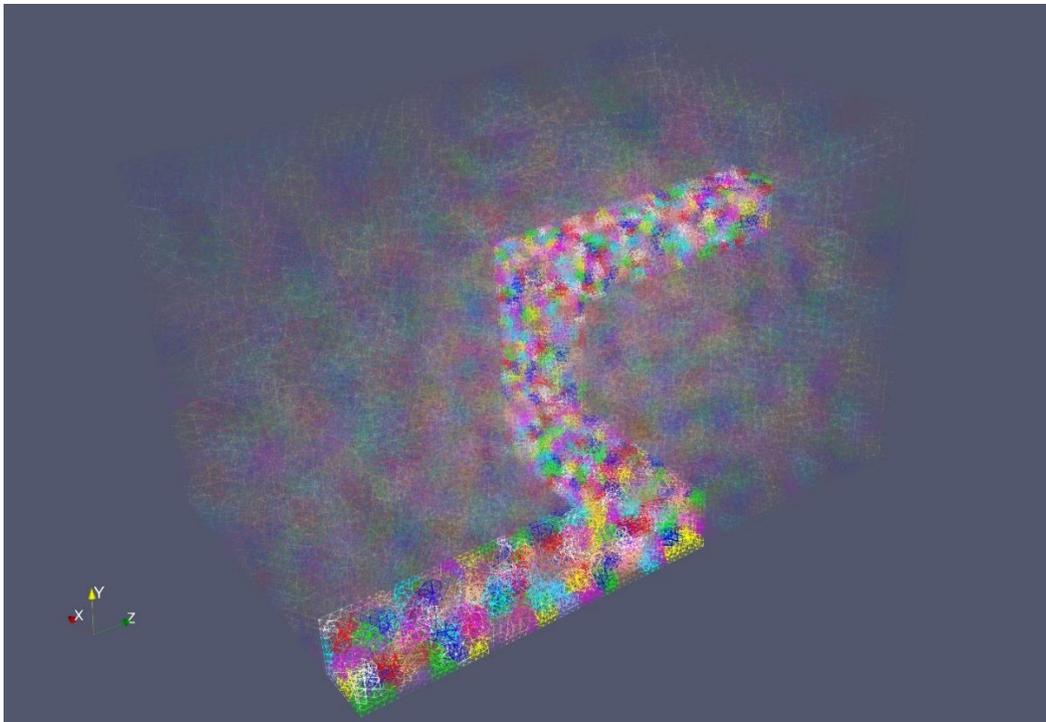


Figure 5.58: Dog-leg void problem interior partitioning

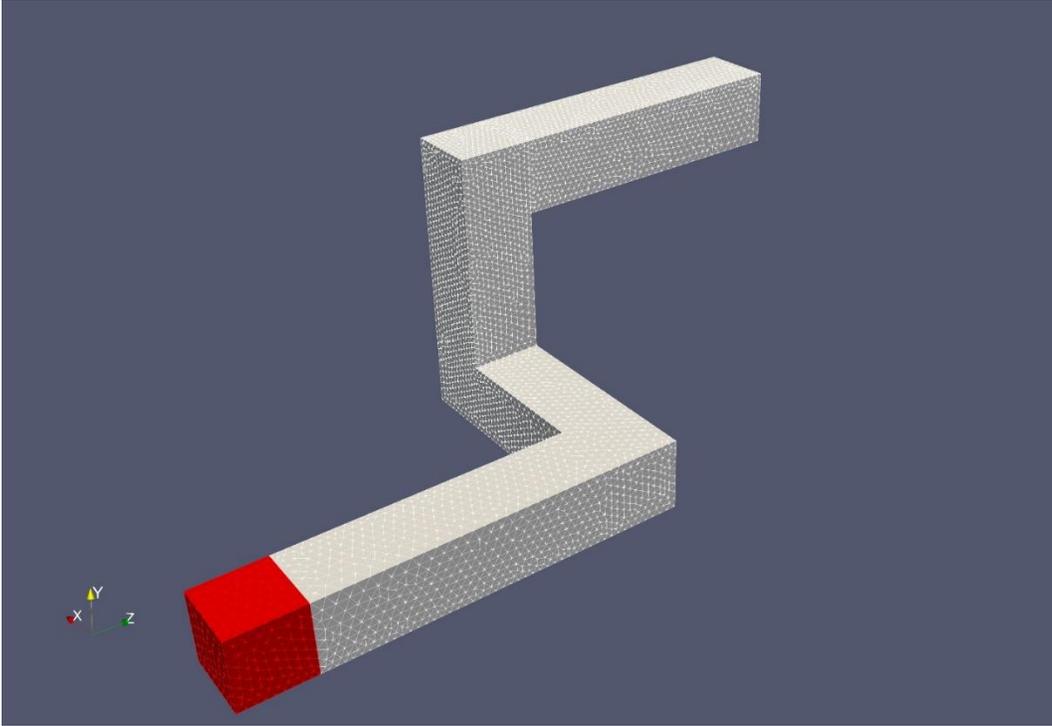


Figure 5.59: Dog-leg void problem duct region mesh

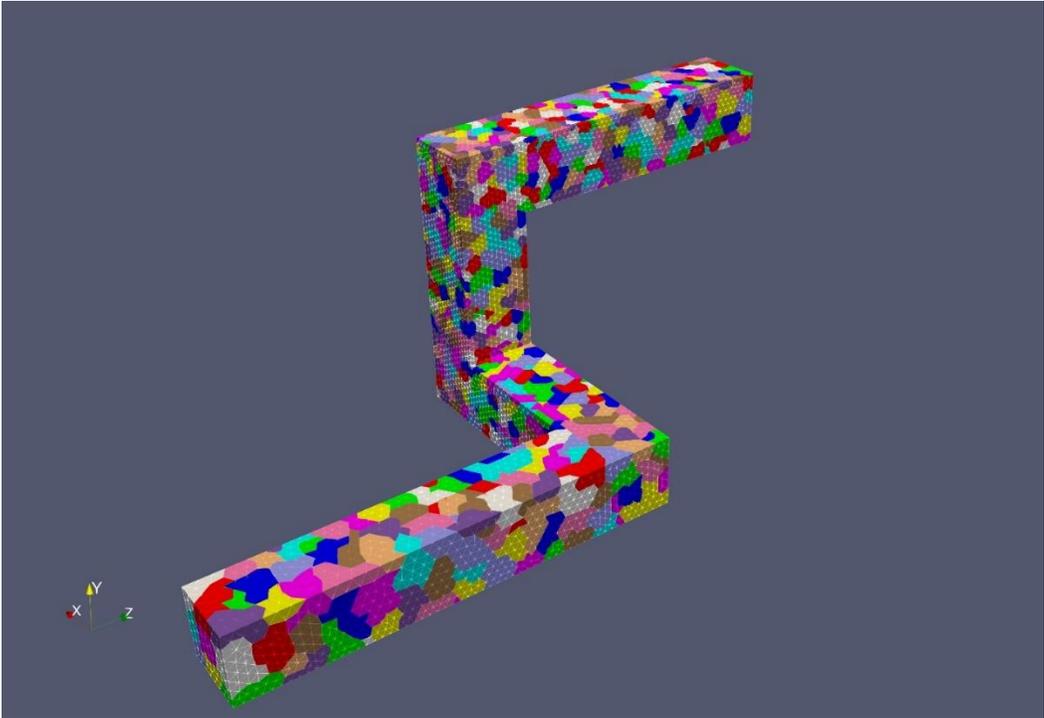


Figure 5.60: Dog-leg void problem duct region partitioning

The total macroscopic cross section of the shielding material is fixed at  $1.0 \text{ cm}^{-1}$  and the relative stopping criteria and angular quadrature are  $10^{-7}$  and  $S_4$ , respectively. Note that this is a single-group, fixed source problem, compared to Godiva and C5G7 which are multi-group,  $k$ -eigenvalue problems. To test the effect of optically thin cells and scattering ratio on the iterative performance of *PBJ-ITMM* and *IPBJ*, we study the test suite created with the set of scattering ratios,  $c \in [0.5, 0.9, 0.98]$  with the duct region either void or filled. “Void” indicates a total cross section of 0.0001 and “filled” indicates a total cross section equal to that of the shielding region. The number of iterations required to converge this problem for the resulting six combinations are displayed in Table 5.11.

Table 5.11: Iterations required by *PBJ-ITMM* and *IPBJ* to converge the dog-leg duct problem with THOR for varying problem parameters

$c$	<i>PBJ-ITMM</i>		<i>IPBJ</i>	
	Void	Filled	Void	Filled
0.5	203	52	222	86
0.9	346	198	539	486
0.98	701	663	1475	1854

The results from this test are consistent with those previously observed. For a given scattering ratio, the filled duct problem converges in fewer iterations than the void duct problem, with the exception of one case. In the most extreme case, the void duct problem requires almost four times more iterations than the filled void problem. This is an effect of the *PBJ* methods being unable to resolve long-distance streaming effectively, a phenomenon studied in depth in previous chapters. For *PBJ-ITMM*, we have demonstrated the ability of a hybrid method, *AH-PI-IP*, to mitigate this iterative slowdown without reducing *PBJ-ITMM*'s degree of parallelism.

As scattering ratio grows in this test, so does the number of required iterations for both the void and filled duct problems. This is because *PBJ* methods also lack robustness with respect to scattering ratio. Additionally, as  $c$  becomes large, the effect of filling the duct has minimal effect on iterative performance. For *IPBJ* in fact, when  $c = 0.98$ , the beneficial effects of leakage outweigh the adverse effects of asynchronicity, and the void duct problem converges in fewer iterations than its filled counterpart. While our hybrid approach was not observed to achieve

robustness with respect to scattering ratio, it still significantly improved *PBJ-ITMM*'s iterative performance when  $c$  grows closer to 1.

This test motivates future implementation of *AH-PI-IP* in THOR for parallel execution on unstructured grids, with the iterative properties previously observed with *PBJ-ITMM* extending to unstructured grids. In this work we have implemented the constituent methods of *AH-PI-IP*, *PBJ-ITMM* and *IPBJ*, as well as our novel algorithm for construction of the *ITMM* matrices, *GFIC*, in THOR. Additionally, tests have demonstrated the individual efficacy of each of these methods, thus providing the necessary framework for future implementation of *AH-PI-IP* in THOR, and more generally on unstructured mesh transport solvers.

## 5.9: Summary of Parallel Numerical Experiments on 3-D Unstructured Grids

The primary objective of this chapter was to utilize the *GFIC* algorithm to implement and test parallel execution of *PBJ-ITMM* on unstructured grids for the first time, along with *IPBJ*, and to execute several numerical experiments and observe performance for comparison. Sections 5.1 - 5.4 detailed the implementation in THOR of the aforementioned methods and algorithm, including discussions of MPI nonblocking communication to avoid deadlock cycles, specifications of the artificial problem setups required by *GFIC*, and a strategy for minimizing RAM utilization during the construction of *ITMM*'s  $K_\psi$  matrix. Measurements of RAM utilization by THOR during execution with *PBJ-ITMM* were then presented in Sec. 5.5, finding the memory requirement to be modest, with the results suggesting memory consumption to be of concern only in problems using very large quadrature orders and/or number of energy groups. For such problems, mitigation strategies were discussed such as a *PBJ-ITMM* implementation solving multiple sub-domains per processor to reduce sub-domain size to decrease the rank of *ITMM* matrices, noting that RAM utilization scales nonlinearly with these ranks.

The bulk of the testing results, strong and weak scaling for the Godiva and C5G7 benchmarks, were presented and discussed in Sec. 5.6. These scaling tests demonstrate *GFIC* to construct the *ITMM* matrices in an expeditious manner, with construction time not comprising a significant portion of the total solution time when the processor count is high. Scaling of the solution times for *PBJ-ITMM* and *IPBJ* as well as their constituent times, pre-process,

computation, and communication demonstrates *PBJ* methods to be more effective at scaling to large problems than excessively parallelizing problems of moderate size. This is a consequence of the asynchronous domain decomposition and the associated increase in number of required iterations as processors are added to a problem of fixed mesh size. When scaling to large problems, however, *PBJ-ITMM* method was observed from weak scaling studies to solve a problem with 6.8 billion unknowns in under 20 minutes, *IPBJ* solving the same problem in under 30 minutes. From the C5G7 weak scaling results, as mesh size and processor count grow larger, communication cost appears to increase by an order of magnitude when the processor count increases by a factor of eight, for sub-domains of constant size and a constant number of required iterations.

With this work demonstrating the first *PBJ-ITMM* algorithm implementation on unstructured grids as an alternative to spatial domain decomposition parallel *SI*, we provide data for a preliminary comparison between these two methods, as well as *IPBJ*. This data, presented in Sec. 5.7, consists of estimated serial execution times consumed by each *PBJ*-type method to solve the Godiva and C5G7 benchmarks on a variety of sub-domain sizes *versus* measured *SI* serial execution times, as well as a comparison of these methods' grind times. The former demonstrates the preferred method from the standpoint of serial execution time to be problem dependent. For Godiva, *SI*'s serial execution time was much shorter than either *PBJ*-type method estimated serial execution time. However, for C5G7, *PBJ-ITMM* executed serially in less time than *SI* for sub-domains between 16 and 64 cells in size, despite requiring more iterations. This data is intended as the starting point for a comparison between *PBJ-ITMM* and *SI* for parallel solution on unstructured grids, with the full comparison requiring a code with capability for parallel solution using both methods. This is suggested as future work, with the implementation strategy and efficacy of *PBJ-ITMM* demonstrated in this work.

*PBJ-ITMM* consuming less serial execution time despite requiring more iterations than *SI* suggested that the observation in HAT-2C of *PBJ-ITMM* having a significantly larger grind time than sweep-based methods was not replicated in THOR, prompting a measurement of THOR's grind times. The resulting data demonstrated *PBJ-ITMM*'s grind time in THOR to be smaller than that of *SI* and *IPBJ* for sub-domains of 128 cells or fewer. This result indicates *PBJ-ITMM*'s utility on unstructured grids, with its matrix-based iterative solution agnostic to mesh type, but also suggests future complications for the *AH-PI-IP* method on unstructured grids. Recall that this method relies on *IPBJ* sub-domains containing more cells than *PBJ-ITMM* sub-domains, while

maintaining a comparable solution time. A full discussion of this consequence is available in Sec. 6.3. We conclude our study of *PBJ-ITMM* and *IPBJ* in parallel execution on unstructured grids with motivation of the future work directed towards implementing *AH-PI-IP* on this platform in Sec. 5.8. This motivation arises from observing the number of required iterations for *PBJ-ITMM* and *IPBJ* to converge a problem containing a large void region, that demonstrates the iterative slowdown previously observed on Cartesian grids.

# Chapter 6:

## Conclusions & Future Work

### 6.1: Conclusions of Hybrid Combinations of *PBJ*-Type and Sweep-Based Methods

The first of our two main objectives in this work was to develop, analyze (Sec. 3.1), and test in serial operation on 2-D Cartesian grids (Chapter 4), hybrid methods to mitigate the iterative slowdown experienced by *PBJ-ITMM* in problems containing optically thin cells while minimizing any penalty to its degree of parallelism on unstructured grids. The simplified configuration of 2-D Cartesian geometry and serial execution was used to expedite the development process. This development began with spectral analysis of both *PBJ-ITMM* and *IPBJ* preconditioned with *SI*, *P-PI-SI* and *P-IP-SI*, respectively. This spectral analysis projected the iterative error onto Fourier basis functions, allowing us to solve for the scaling (eigenvalues) of the iterative error in the cell-averaged scalar flux (eigenmodes) as a function of the Fourier variables. The fundamental mode was determined to coincide with the origin in Fourier space, indicating the slowest converging error mode (Sec. 4.2). The spectral radius trends produced by this analysis indicated that the preconditioning approach achieved iterative robustness with respect to optical thickness for  $c < 1$ , but that using only one of the two constituent methods of the hybrid scheme would be more advantageous than the preconditioning combination for most homogeneous, uniformly meshed problems. This was further supported by observing the number of iterations consumed by *SI*, *PBJ-ITMM*, *IPBJ*, *P-PI-SI*, and *P-IP-SI* as implemented in our HAT-2C code to converge a homogeneous problem with varying nuclear properties (Sec. 4.4).

These findings led to the development of the *AH-PI-SI* and *AH-IP-SI* methods, as it was realized that in a heterogeneous problem (Sec. 4.5) containing cells of varying optical thickness, one constituent method was likely sufficient for a given cell's solution convergence, with little assistance from the other method. The appropriate method for a given cell is determined by its optical thickness, with the *PBJ*-type and sweep-based methods solving cells with optical thicknesses above and below a prescribed value, respectively. The *AH* approach executes in this

manner, with lagged angular fluxes on zone interfaces, where zones are defined as contiguous collections of cells in a common thin or thick regime. The fundamental mechanism for this technique is effectively modeling the long-distance streaming in optically thin cells and the localized collision in optically thick cells using *SI* and *PBJ-ITMM*, respectively. Compared to the preconditioning counterparts, the *AH* approach reduces the total number of operations required per iteration, with only one method being executed in each cell, and reduces the penalty to *PBJ*'s degree of parallelism due to the elimination of the global mesh sweep required by *P-PI-SI* and *P-IP-SI*. Testing on a heterogeneous periodic vertical interface problem demonstrated *AH-PI-SI* to require only a small increase in iterations compared to *P-PI-SI*. *AH-IP-SI*, however, required significantly more iterations than *P-IP-SI*, with *IPBJ* unable to effectively resolve localized collisions and its convergence rate being, under the best-case-scenario, equal to that of *SI*. This study therefore suggests the *AH* approach to be far more advantageous for practical implementation than the preconditioning approach, and *PBJ-ITMM* to be better suited as the primary iterative method than *IPBJ* in this configuration.

Following this development, we conducted a parametric study (Sec. 4.6 and 4.7), varying the number of cells per stripe, scattering ratio, and cell thicknesses of the aforementioned periodic vertical interface problem, measuring the required iterations and execution time for various methods. In addition to the hybrid methods discussed thus far, this study implements the final hybrid method developed in this work, *AH-PI-IP*. This method executes *IPBJ* in optically thin regions instead of *SI*, utilizing the linear scaling of *IPBJ*'s iterative solution time to decompose thin regions into sub-domains containing cells in excess of *PBJ-ITMM*'s practical limit. For our study specifically, we prescribed *IPBJ* sub-domains in the *AH-PI-IP* implementation to be of a size such that their per-angle iterative solution time does not exceed that of *PBJ-ITMM*'s per sub-domain solution time. Since angularly parallel sweeps on unstructured grids do not impose the scheduling complexities to which we propose *PBJ* methods as an alternative, *AH-PI-IP* with this prescribed size limit on *IPBJ* sub-domains imposes no penalty to *PBJ-ITMM*'s degree of parallelism. Including three traditional acceleration schemes, the parametric study tests and contrasts the iterative performance of 11 iterative methods, *SI*, *DSA*, *AP*, *pNDA*, *PBJ-ITMM*, *IPBJ*, *P-PI-SI*, *P-IP-SI*, *AH-PI-SI*, *AH-IP-SI*, and *AH-PI-IP* over a wide range of problem parameters.

The results of this parametric study demonstrate *PBJ-ITMM* to definitively be preferable to *IPBJ* as the primary method in our hybrid approach, with both *P-IP-SI* and *AH-IP-SI* providing

little acceleration to *IPBJ* as the scattering ratio increases. The hybrid approach is found to be effective, however, with *PBJ-ITMM* as the primary method, reducing the number of required iterations compared to *PBJ-ITMM*, and also being far less sensitive to scattering ratio than *IPBJ* or any hybrid approach using it as the primary method. While iterative robustness with respect to scattering ratio was neither expected nor observed, the *AH-PI-SI* and *AH-PI-IP* provided acceleration to *PBJ-ITMM* as the scattering ratio increased, even converging certain problems when  $c = 1$  in under 1000 iterations. In general, the improvement of *PBJ-ITMM*'s convergence rate from the hybrid approach was maximized in problems containing large regions of very optically thin cells. While *AH-PI-IP* was observed to typically result in only a modest increase in the number of required iterations compared to *AH-PI-SI*, when the number of cells per stripe was large, the difference increased, with *IPBJ* imposing asynchronicity into these large, optically thin regions. This fact notwithstanding, these are also the cases in which *AH-PI-SI* imposes the largest penalty to *PBJ-ITMM*'s degree of parallelism, which *AH-PI-IP* eliminates. Comparing to traditional acceleration methods, *DSA*, *AP*, and *pNDA* all converge in fewer iterations than our hybrid approaches for almost all cases. However, since these acceleration methods are not readily extendible to massively parallel solution on unstructured grids, we view the observed hybrid execution times that are typically within a factor of ten of those observed for the acceleration methods favorably.

Finally, in Sec. 4.8, we tested our hybrid approaches on realistic test problems containing void regions. The previously predicted and observed trends from analysis and controlled testing were all found to extend to these realistic configurations. The final result of this portion of our work is to establish that *AH-PI-IP* is able to reduce, in some cases greatly, the number of iterations required by *PBJ-ITMM* to converge problems containing optically thin cells without reducing the method's degree of parallelism on unstructured grids.

## 6.2: Conclusions of *GFIC*, *PBJ-ITMM*, & *IPBJ* in Parallel Execution on Unstructured Grids

While the algorithms necessary for implementing *IPBJ* on unstructured grids previously existed, *PBJ-ITMM* had never been implemented for such configurations due to the lack of an algorithm for construction of the associated matrices. This leads to the second primary objective of this work, development of an algorithm for construction of the *ITMM* matrices on unstructured grids and its implementation for parallel solution. For this task, we developed the *GFIC* algorithm which constructs the *ITMM* matrices using the pre-existing mesh sweep algorithm typically implemented in  $S_N$  transport codes. Formulated in Sec. 3.2.4, *GFIC* utilizes the interpretation of *ITMM* as a response matrix method to construct the associated matrices by creating an artificial problem, a single *SI* iterative solution of which produces the desired responses, and hence, composes the *ITMM* matrix elements. This approach is preferable to the development of a new kernel calculation for the earlier *DMS* algorithm on unstructured grids because it allows for implementation in  $S_N$  transport codes utilizing the current mesh sweep and kernel calculation. Consequently, *GFIC* does not require re-derivation for different spatial discretization methods, and also imposes a reduced amount of additional code maintenance, as upgrades and new capabilities added to the basic *SI* solver are expected to largely extend to the *PBJ-ITMM* solution routines by virtue of the matrices being constructed using the standard *SI* routines. *PBJ-ITMM* enabled by *GFIC*, as well as *IPBJ* were both implemented into the THOR code for parallel execution on unstructured grids using nonblocking MPI communication, as detailed in Sec. 5.1 - 5.4.

Following this implementation, our objective was to verify to efficacy of *GFIC* and to then test the enabled *PBJ-ITMM* solution in parallel execution for a set of problems with unstructured tetrahedral grids (Sec. 5.6). The main conclusions from these scaling studies pertain to the relative performance of *PBJ-ITMM* and *IPBJ* as the number of processors used for execution increases. In general, strong scaling results suggest that both methods are ineffective for excessively parallelizing a problem, i.e. continually adding processors to solve a problem of fixed size. Due to the increase in the iteration count as sub-domain size decreases, the benefit of adding processors becomes outweighed by the degradation of iterative performance when sub-domains become

small. The primary finding from our strong scaling results was that *PBJ-ITMM* executes faster than *IPBJ* when the number of cells per sub-domain is below a problem-dependent threshold. This threshold was observed to be higher in the more computationally difficult problem tested, C5G7. This indicates that the range of sub-domain sizes over which *PBJ-ITMM* is preferable increases for problems that are most in need of parallelization, increasing its utility.

Weak scaling tests demonstrated both methods to be well suited for solving large problems in the massively parallel regime. The Godiva and C5G7 benchmark problems meshed with over eight million cells, with six and seven group respectively, and  $S_4$  quadrature were found to converge using *PBJ-ITMM* to a relative stopping criterion of  $10^{-6}$  in under 5 and 20 minutes, respectively, when executed on 32,768 processors. The latter executes in under 30 minutes with *IPBJ*, demonstrating a significant improvement with *PBJ-ITMM* on the largest problem solved in this work.

Given *PBJ*'s asynchronicity, weak scaling performance was found to vary depending on whether the added cells to a mesh resulted in cells with about the same optical thickness (i.e. addition of problem volume) or in cells that are substantially thinner (i.e. mesh refinement). Godiva was weak scaled using mesh refinement, which resulted in an increase in the number of required iterations as processor count increased, thus increasing the slope of the scaling trend. C5G7, on the other hand, increased the problem volume proportionately with processor count and, consequently, had very little variation in the number of required iterations over a single weak scaling trend. Some of these trends were observed to enter the asymptotic regime at high processor counts, the slope of which indicates that the communication time associated with both methods increases by roughly an order of magnitude when the processor count is increased by a factor of eight.

With the ability of *PBJ-ITMM* and *IPBJ* to scale to very large problems demonstrated, we have compared the serial operation execution times of *PBJ-ITMM*, *IPBJ*, and *SI*, as well as their associated grind times in THOR (Sec. 5.7). These studies indicate the faster method between *PBJ-ITMM* and *SI* in terms of serial execution time to be heavily problem dependent. This comparison ultimately provides the starting point for a future parallel comparison between *SI* and *PBJ-ITMM* on unstructured grids. Additionally, the grind time for *PBJ-ITMM* in THOR is determined to be less than the grind time of sweep-based methods *SI* and *IPBJ* when sub-domains are comprised of 128 or fewer cells. Finally, testing *PBJ-ITMM* and *IPBJ* in parallel on the dog-leg duct benchmark

problem (Sec. 5.8) demonstrates the iterative slowdown of *PBJ*-type methods in problems containing optically thin cells, motivating the future implementation of *AH-PI-IP* in THOR.

The primary conclusion from our studies on 3-D unstructured grids is that our novel algorithm *GFIC* was able to produce the first *PBJ-ITMM* solver on unstructured grids, thus allowing this method to be studied in the configuration to which it has long been thought to be most applicable. *GFIC* was demonstrated in this work to be a highly practical algorithm for this task, constructing the required matrices in a time that, in the massively parallel regime, is short compared to the overall execution time. Additionally, its versatility was demonstrated, with the implementation not requiring us to fundamentally alter THOR, but rather, simply impose alternate problem specifications. With *GFIC* enabling *PBJ-ITMM* solution on unstructured grids, the ensuing scaling tests have demonstrated *PBJ-ITMM* to be a viable option for massively parallel solution on unstructured grids, providing an alternative to spatially parallelized *SI*.

### 6.3: Proposed Future Work

The primary suggestion for future work is the implementation of *AH-PI-IP* in THOR for parallel execution. Prior to this implementation, however, enhancements to the *PBJ-ITMM* capabilities in THOR are recommended. Firstly, the capability of executing multiple sub-domains per processor is suggested to be advantageous, both for *PBJ-ITMM* itself, as well as the ensuing *AH-PI-IP* implementation. Our weak scaling results have demonstrated *PBJ-ITMM* (and *IPBJ*) to be heavily communication dominated at high processor counts. With computation comprising such a small portion of the solution time, execution of multiple sub-domains each iteration on a common processor is likely advantageous, as it decreases the total amount of data that must be communicated each iteration. Optimal implementation of this strategy would require a two-step decomposition, decomposing first into processor domains, then decomposing these processor domains into the constituent sub-domains. This process ensures that the sub-domains a single processor is responsible for solving comprise a contiguous region.

In addition to potentially improving the performance of *PBJ-ITMM*, this implementation would also likely improve the performance of *AH-PI-IP*. In Sec 5.7, it was determined that *PBJ-ITMM*'s grind time on unstructured grids was comparable to or shorter than *IPBJ*'s for practical *PBJ-ITMM* sub-domain sizes (up to ~256 cells). This produces complications for *AH-PI-IP*, as

this method relies on *IPBJ* sub-domains with more cells than, but comparable iterative solution times, to *PBJ-ITMM* sub-domains. With *PBJ-ITMM*'s grind time being much closer to, and in many cases shorter than, *IPBJ*'s on unstructured grids, accomplishing this goal becomes more difficult. Angular parallelism of *IPBJ*'s solution was previously suggested to increase *IPBJ* sub-domain sizes and this option is still both available and recommended. Solving multiple *PBJ-ITMM* sub-domains per processor would further increase the allowable *IPBJ* sub-domain size.

Another parallelism technique recommended for study is first decomposing, not into processor domains, but into node domains, then decomposing further into sub-domains. The sub-domains would then be solved on a node using shared memory parallelism. Since shared memory parallelism is very cheap in comparison to off-node communication, this approach would possibly decrease the communication cost by ensuring that processors on a common node solve a contiguous group of sub-domains. The final *PBJ-ITMM* implementation technique we propose for future studies is to execute multiple iterations on the sub-domains assigned to a given processor (or node) before communicating. This would allow these larger processor or node domains to resolve some of the internal asynchronicity without communication costs.

After these *PBJ-ITMM* implementations are considered, the next logical step in the development process would be to implement *AH-PI-IP* in THOR. This would require a two-step decomposition similar to that described for the proposed *PBJ-ITMM* adaptations. The first step would decompose the domain into partitions of a specified *IPBJ* sub-domain size. A calculation would then be used to assess the optical thickness of these sub-domains. On unstructured grids, this is not trivial as it is on Cartesian grids. A proposed procedure for this calculation is to multiple the total cross section in each cell by that cell's volume then sum over all cells in the partition. Another alternative would be to ray trace along the quadrature's discrete ordinates from the volumetric center of a partition to the partition boundary. The average number of path lengths traversed during this process may then be used as an estimate of a partition's optical thickness. The former of these procedures is less computationally expensive, as it does not require ray tracing, however, the latter would provide a measurement of optical thickness that accounts for sub-domain shape. If the value computed by the chosen approach is below a prescribed threshold, this partition becomes an *IPBJ* sub-domain and would be assigned to a node for shared memory angularly parallel solution. Otherwise, this partition is to be solved with *PBJ-ITMM* and is decomposed with either the currently implemented strategy or one of the aforementioned adaptations. Additionally,

since cross sections vary with energy group, the partitioning would perform best if adapted to be group-dependent.

# References

- [1] Duderstadt, J. J., & Hamilton, L. J. (1976). *Nuclear Reactor Analysis*. New York: John Wiley & Sons.
- [2] Lewis, E. E., & Miller, W. F. (1993). *Computational Methods of Neutron Transport*. La Grange Park, Illinois USA: American Nuclear Society, Inc.
- [3] Nunes, C., Barros, R. C., & Filho, H. A. (2013). Albedo Boundary Conditions for Thermal Nuclear Reactor Global Calculations with Two-Energy Group Discrete Ordinates Model. International Nuclear Atlantic Conference.
- [4] Azmy, Y. Y., Anistratov, D., & Zerr, R. J. (2015). Numerical and Analytical Studies of the Spectrum of Parallel Block Jacobi Iterations for Solving the Weighted Diamond Difference Form of the Sn Equations. Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) method, Nashville, TN, USA.
- [5] Azmy, Y. Y. (1988). The Weighted Diamond-Difference Form of Nodal Transport Methods. *Nuclear Science and Engineering*, 98(1), 29–40.
- [6] Adams, M. L, Larsen, E. W. (2002). Fast Iterative Methods for Discrete-Ordinates Particle Transport Calculations. *Progress in Nuclear Energy*, 40(1), 3–159.
- [7] Larsen, E. W. (1982). Unconditionally Stable Diffusion-Synthetic Acceleration Methods for the Slab Geometry Discrete Ordinates Equations. Part 1: Theory. *Nuclear Science and Engineering*, 82, 47–63.
- [8] Kopp, H. J. (1963). Synthetic Method Solution of the Transport Equation. *Nuclear Science and Engineering*, 17(1), 65–74.

- [9] Reed, W. H. (1971). The Effectiveness of Acceleration Techniques for Iterative Methods in Transport Theory. *Nuclear Science and Engineering*, 45(3), 245–254.
- [10] Gelbard, E. M., & Hageman, L. A. (1969). The Synthetic Method as Applied to the Sn Equations. *Nuclear Science and Engineering*, 37, 288–298.
- [11] Alcouffe, R. E. (1977). Diffusion Synthetic Acceleration Methods for the Diamond-Differenced Discrete-Ordinates Equations. *Nuclear Science and Engineering*, 64, 344–355.
- [12] Azmy, Y. Y., & Dorning, J. J. (1985). Diffusion Synthetic Acceleration of the Multi-Dimensional Discrete Nodal Transport Method. *Advances in Nuclear Engineering Computational Methods*, 2, 440–451.
- [13] Azmy, Y. Y. (1985). Nodal Methods for Problems in Fluid Mechanic and Neutron Transport. PhD Dissertation: University of Illinois at Urbana-Champaign.
- [14] Azmy, Y. Y., & Larsen, E. W. (1987). Fourier Analysis of the Diffusion Synthetic Acceleration Method for Weighted Diamond Differencing Schemes in Cartesian Geometries. *Nuclear Science and Engineering*, 95(2), 106–115.
- [15] Yamamoto, T. (1995). Optimization Study on the Diffusion Synthetic Acceleration Algorithm in Three-Dimensional Discrete Ordinate Transport. *Journal of Nuclear Science and Technology*, 32(8), 804–812.
- [16] Muhammad, H., & Hong, S. G. (2018). Effective Use of the Linear Fine Mesh Rebalance for the DSA of  $S_N$  Transport Equation with Tetrahedral Meshes. *Transactions of the American Nuclear Society*, 118, 376–379.
- [17] Févotte, F. (2018). Piecewise Diffusion Synthetic Acceleration scheme for neutron transport simulations in optically thick systems. *Annals of Nuclear Energy*, 118, 71–80.

- [18] Azmy, Y. Y. (1993). Cell-Centered Imposed Diffusion Synthetic Acceleration for Weighted Difference Transport Methods. *Nuclear Science and Engineering*, 115(3), 265–272.
- [19] Azmy, Y. Y. (1999). Iterative Convergence Acceleration of Neutral Particle Transport Methods via Adjacent-Cell Preconditioners. *Journal of Computational Physics*, 152, 359–384.
- [20] Azmy, Y. Y. (2000). Acceleration of Multidimensional Discrete Ordinates Methods Via Adjacent-Cell Preconditioners. *Nuclear Science and Engineering*, 136, 202–226.
- [21] Rosa, M., Azmy, Y. Y., & Morel, J. E. (2009). Properties of the Sn-Equivalent Integral Transport Operator in Slab Geometry and the Iterative Acceleration of Neutral Particle Transport Methods. *Nuclear Science and Engineering*, 162(3), 234–252.
- [22] Azmy, Y. Y. (2002). Unconditionally Stable and Robust Adjacent-Cell Diffusive Preconditioning of Weighted-Difference Particle Transport Methods is Impossible. *Journal of Computational Physics*, 182, 213–233.
- [23] Warsa, J. S., Wareing, T. A., & Morel, J. E. (2004). Krylov Iterative Methods and the Degraded Effectiveness of Diffusion Synthetic Acceleration for Multidimensional SN Calculations in Problems with Material Discontinuities. *Nuclear Science and Engineering*, 147(3), 218–248.
- [24] Cefus, G. R., & Larsen, E. W. (1990). Stability Analysis of Coarse-Mesh Rebalance. *Nuclear Science and Engineering*, 105(1), 31–39.
- [25] Park, Y. R., & Cho, N. Z. (2008). Coarse-Mesh Angular Dependent Rebalance Acceleration of the Method of Characteristics in x-y Geometry. *Nuclear Science and Engineering*, 158(2), 154–163.

- [26] Gol'din, V. Y. (1964). A Quasi-Diffusion Method of Solving the Kinetic Equation. USSR Computational Mathematics and Mathematical Physics, 4(6), 136–149.
- [27] Anistratov, D. Y., & Gol'din, V. Y. (2011). Multilevel Quasidiffusion Methods for Solving Multigroup Neutron Transport k-Eigenvalue Problems in One-Dimensional Slab Geometry. Nuclear Science and Engineering, 169(2), 111–132.
- [28] Cornejo, L. R., & Anistratov, D. Y. (2017). The multilevel quasidiffusion method with multigrid in energy for eigenvalue transport problems. Progress in Nuclear Energy, 101, 401–408.
- [29] Wieselquist, W. A., Anistratov, D. Y., & Morel, J. E. (2014). A cell-local finite difference discretization of the low-order quasidiffusion equations for neutral particle transport on unstructured quadrilateral meshes. Journal of Computational Physics, 273, 343–357.
- [30] Smith, K. S., & Rhodes, J. D. (2002). Full-Core, 2-D LWR Core Calculations with CASMO-4E. PHYSOR.
- [31] Park, H., Knoll, D. A., & Newman, C. K. (2017). Nonlinear Acceleration of Transport Criticality Problems. Nuclear Science and Engineering, 172(1), 52–65.
- [32] Cornejo, L. R., & Anistratov, D. Y. (2016). Nonlinear Diffusion Acceleration Method with Multigrid in Energy for k-Eigenvalue Neutron Transport Problems. Nuclear Science and Engineering, 184, 514–526.
- [33] Anistratov, D. Y. (2013). Multilevel NDA Methods for Solving Multigroup Eigenvalue Neutron Transport Problems. Nuclear Science and Engineering, 174, 150–162.
- [34] Xiao, S., Ren, K., & Wang, D. (2018). A Local Adaptive Coarse-Mesh Nonlinear Diffusion Acceleration Scheme for Neutron Transport Calculations. Nuclear Science and Engineering, 189(3), 272–281.

- [35] Cho, N. Z. (2005). Fundamentals and Recent Developments of Reactor Physics Methods. *Nucl. Sci. & Eng.*, 37(1), 25–78.
- [36] Ramone, G. L., Adams, M. L., & Nowak, P. F. (1997). A Transport Synthetic Acceleration Method for Transport Iterations. *Nuclear Science and Engineering*, 125(3), 257–283.
- [37] Azmy, Y. Y. (1997). Multiprocessing for Neutron Diffusion and Deterministic Transport Methods. *Progress in Nuclear Energy*, 31(3), 317–368.
- [38] Yessayan, R. A. (2018). Improvements to the THOR Neutral Particle Transport Code on High-Performance Computing Systems via Acceleration, Parallelization, and Performance Analysis. MS Thesis: North Carolina State University.
- [39] Boyd, W., Siegel, A., He, S., Forget, B., & Smith, K. (2016). Parallel Performance Results for the OpenMOC Neutron Transport Code on Multicore Platforms. *International Journal of High Performance Computing Applications*, 30(3), 360–375.
- [40] Ferrer, R. M., & Azmy, Y. Y. (2012). A Robust Arbitrarily High-Order Transport Method of the Characteristic Type for Unstructured Grids. *Nuclear Science and Engineering*, 172(1), 33–51.
- [41] Ferrer, R. M. (2010). An Arbitrarily High Order Transport Method of the Characteristic Type for Unstructured Tetrahedral Grids. PhD Dissertation: Penn State University.
- [42] Kock, K. R., Baker, R. S., & Alcouffe, R. E. (1992). Solution of First-Order Form of Three-Dimensional Discrete Ordinates Equations on a Massively Parallel Machine. *Transactions of the American Nuclear Society*, 65, 198.
- [43] Fischer, J. W., & Azmy, Y. Y. (2007). Comparison via parallel performance models of angular and spatial domain decompositions for solving neutral particle transport problems. *Progress in Nuclear Energy*, 49, 37–60.

- [44] Alcouffe, R. E., *et al.* (2018). PARTISN: A Time-Dependent, Parallel Neutral Particle Transport Code System. Computational Physics and Methods, CCS-2, Los Alamos National Laboratory, Version 8.29.
- [45] Azmy, Y. Y. (1997). Multiprocessing for Neutron Diffusion and Deterministic Transport Methods. *Progress in Nuclear Energy*, 31(3), 317–368.
- [46] Pautz, S. D. An Algorithm for Parallel Sn Sweeps on Unstructured Meshes. *Nuclear Science and Engineering*, 140(2), 111–136.
- [47] Plimpton, S., Hendrickson, B., Burns, S., & McLendon, W. (2000). Parallel Algorithms for Radiation Transport on Unstructured Grids. Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, Dallas, TX, USA.
- [48] Kumar, V. S. A., Marathe, M. V., Parthasarath, S., Srinivasan, A., & Zust, S. (2006). Provable Algorithms for Parallel Generalized Sweep Scheduling. *Journal of Parallel and Distributed Computing*, 66, 807–821.
- [49] Pautz, S. D., & Bailey, T. S. (2016). Parallel Deterministic Transport Sweeps of Structured and Unstructured Meshes with Overloaded Mesh Decompositions. *Nuclear Science and Engineering*, 185(1).
- [50] Adams, M. P. (2020). Provably Optimal Parallel Transport Sweeps on Semi-Structured Grids. *Journal of Computational Physics*, 407.
- [51] Yavuz, M., & larsen, E. W. (1989). Spatial domain decomposition for Neutron transport problem. *Transport Theory and Statistical Physics*, 18(2), 205–219.
- [52] Zerr, R. J. (2011). Solution of the Within-Group Multidimensional Discrete Ordinates Transport Equations on Massively Parallel Architectures. PhD Dissertation: Penn State University.

- [53] Azmy, Y. Y. (1997). A New Algorithm for Generating Highly Accurate Benchmark Solutions to Transport Test Problems. *Proceedings XI ENFIR/IV ENAN Joint Nuclear Conferences, August 18-22, 1997, Pocos de Caldas Springs, MG, Brazil.*
- [54] Hanebutte, U. R., & Lewis, E. E. (1992). A Massively Parallel Discrete Ordinates Response Matrix Method for Neutron Transport. *Nuclear Science and Engineering*, 111(1), 45–56.
- [55] Powell, B. P. (2013). An Advanced Algorithm for Construction of Integral Transport Matrix Method Operators Using Accumulation of Single Cell Coupling Factors. MS Thesis: North Carolina State University.
- [56] Hoagland, D. S., Azmy, Y. Y., & Zerr, R. J. (2016). A Study of the Iterative Convergence of the Integral Transport Matrix Method in Two-Dimensional Geometry for the Diamond Difference Method. *Physor 2016: Unifying Theory and Experiments in the 21<sup>st</sup> Century*, Sun Valley, ID.
- [57] Rosa, M., Warsa, J. S., & Chang, J. H. (2010). Fourier Analysis of Inexact Parallel Block-Jacobi Splitting with Transport Synthetic Acceleration. *Nuclear Science and Engineering*, 164, 248–263.
- [58] Hoagland, D. S., & Azmy, Y. Y. (2017). Iterative Properties of Parallel Block Jacobi - Integral Transport Matrix Method with Source Iteration Preconditioning. *International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering*, Jeju, Korea.
- [59] Hoagland, D. S., & Azmy, Y. Y. (2019). Fourier Analysis of Inexact Parallel Block Jacobi with Source Iteration Preconditioning and Computational Experimentation with Parallel Block Jacobi / Source Iteration Hybrid Methods. *International Conference on Mathematics & Computational Methods Applied to Nuclear Science and Engineering*, Portland, Oregon.

- [60] Hoagland, D. S., & Azmy, Y. Y. (2020). Parametric Study of Parallel Block Jacobi / Source Iteration Hybrid Methods in 2-D Cartesian Geometry and Construction of the Integral Transport Matrix Method Matrices via Green's Functions. *PHYSOR 2020: Transition to a Scalable Nuclear Future*, Cambridge, United Kingdom.
- [61] Song, P., Zhang, Z., Zhang, Q., Liang, L., & Zhao, Z. (2020). Implementation of the CPU/GPU Hybrid Parallel Method of Characteristics Neutron Transport Calculation Using the Heterogeneous Cluster with Dynamic Workload Assignment. *Annals of Nuclear Energy*, 135.
- [62] Wolfram Research, Inc., Mathematica, Version 11.3, Champaign, IL (2018).
- [63] Anderson, E., *et al.* (1999). *LaPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics.
- [64] Greenbaum, A., & Seager, M. DLAP Sparse Linear Algebra Package.
- [65] Kaasschieter, E. (1986). The Solution of Non-Symmetric Linear Systems by Bi-Conjugate Gradients or Conjugate Gradients Squared. *Delft University of Technology Report*, 86(21).
- [66] MathWorks, Inc. (1996). *MATLAB: the language of technical computing: computation, visualization, programming: installation guide for UNIX version 5*. Natick: Math Works Inc.
- [67] Kobayashi, K., Sugimura, N., & Nagaya, Y. (2000). 3-D Radiation Transport Benchmark Problems and Results for Simple Geometries with Void Regions. *Nuclear Energy Agency Organization for Economic Co-Operation and Development*.
- [68] Geuzaine, C., & Remacle, J. F. (2009). Gmsh: A Three-Dimensional Finite Element Mesh Generator with Build-in Pre- and Post-processing Facilities. *International Journal for Numerical Methods in Engineering*, 79(11), 1309–1331.

- [69] Karypis, G., & Kumar, V. (1999). A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1), 359–392.
- [70] Idaho National Laboratory: High Performance Computing. Retrieved from [hpc.inl.gov](http://hpc.inl.gov)
- [71] Intercomparison of Calculations for Godiva and Jezebel: An Intercomparison study by the JEFF Project with contributions from Britain, France, the Netherlands, and Switzerland. (1999). *Nuclear Energy Agency Organisation for Economic Co-Operation and Development*.
- [72] Benchmark on Deterministic Transport Calculations Without Spatial Homogenisation: A 2-D/3-D MOX Fuel Assembly Benchmark. (2003). *Nuclear Energy Agency Organisation for Economic Co-Operation and Development*.
- [73] Ayachit, & Utkarsh. (2015). *The ParaView Guide: A Parallel Visualization Application*. Kitware.

# Appendices

## Appendix A: Full Results from Parametric Study in Section 4.6

This appendix contains the full data set collected in the parametric study presented in Section 4.6. To recap, this parametric study was conducted on a  $128 \times 128$  mesh of the heterogeneous stripe problem depicted in Fig. 4.23, with an increased mesh size and a variable number of stripes and cell thicknesses. The specifications and conclusions for this study are available in Section 4.6. The following legend in Fig. A.1 applies to all graphs within this appendix. For ease of viewing and compactness, this legend will be omitted from all graphs. The legend is color coordinated such that all acceleration methods are varying shades of green, all methods containing *PBJ-ITMM* or *IPBJ* are shades of blue or red respectively (except for *AH-PI-IP*). The *PBJ* method by itself is the darkest shade, with the *SI* preconditioned method a lighter shade, and the *asynchronous hybrid* combination with *SI* a lighter shade still. Additionally, the captions below each graph are color coded to easily discern which parameters have changed between cases. Recall that there are three parameters that vary between cases, number of cells per stripe,  $N_s$ , and the optical thickness of thin and thick cells,  $\Sigma_{thin}\Delta u_{thin}$  and  $\Sigma_{thick}\Delta u_{thick}$  respectively. The ranges of these variables are,  $N_s = [4, 8, 16, 32, 64]$ ,  $\Sigma_{thin}\Delta u_{thin} = [0.001, 0.01, 0.1, 0.5]$ , and  $\Sigma_{thick}\Delta u_{thick} = [1, 2.5, 5, 10]$ , with the displayed color coding. Recall that for this study, we refer to  $N_s = 4$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ , and  $\Sigma_{thick}\Delta u_{thick} = 5$  as the “base case”. Any value decreased two data points below the base case are colored **dark red**; one data point below, **light red**; one data point above, **light blue**; and two data points above, **dark blue**. These color codings are to allow the large amount of data from the parametric study to be interpreted efficiently. This color coding is also applicable to Appendix B.

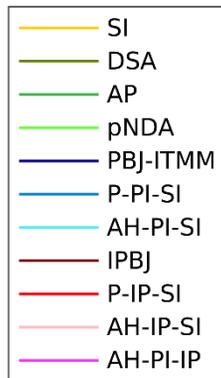


Figure A.1: Legend for all subsequent graphs in current appendix

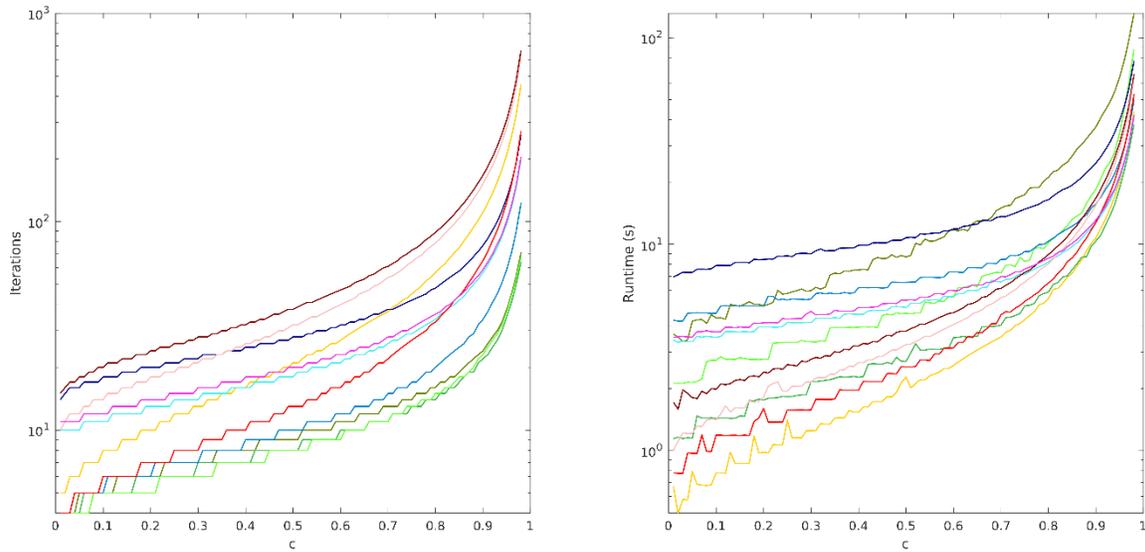


Figure A.2: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 1$

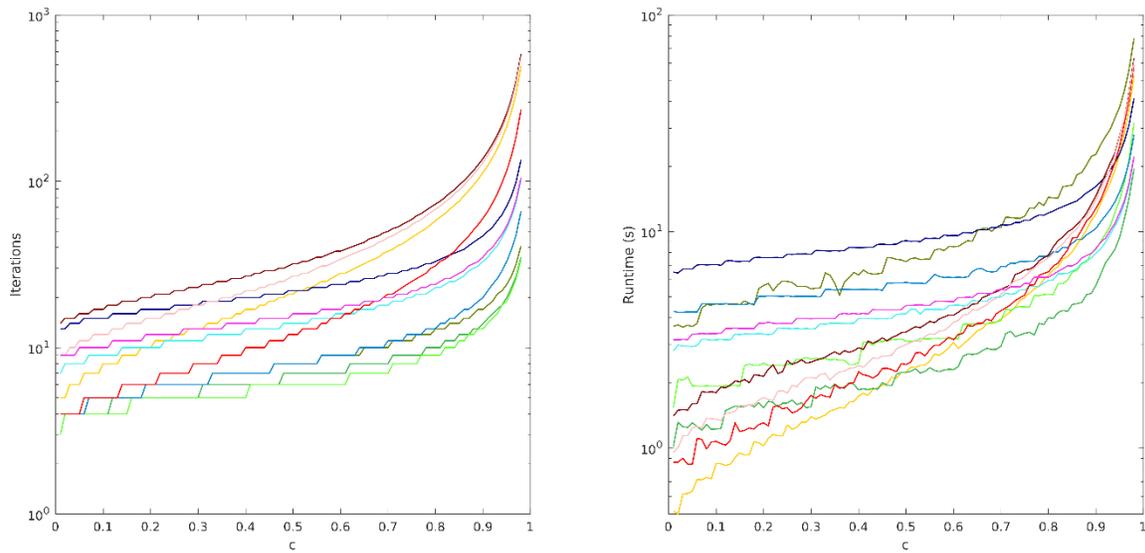


Figure A.3: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 2.5$

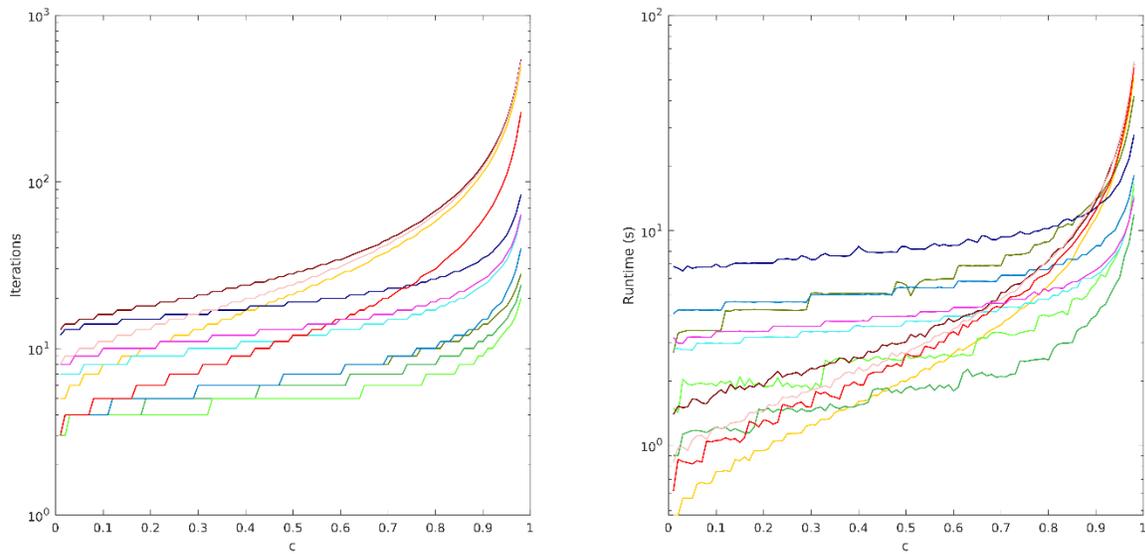


Figure A.4: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin} \Delta u_{thin} = 0.001, \Sigma_{thick} \Delta u_{thick} = 5$

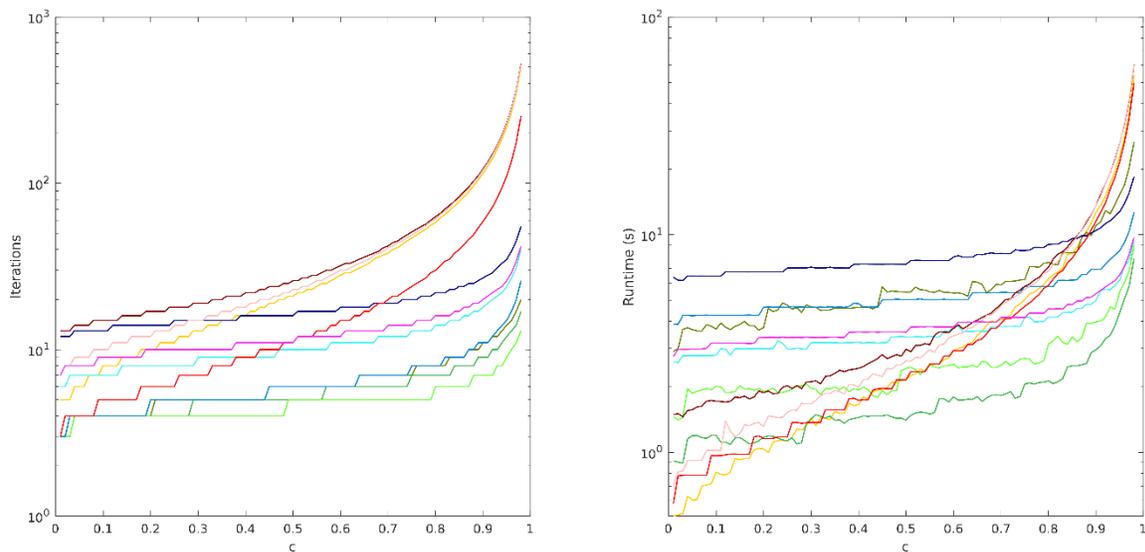


Figure A.5: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin} \Delta u_{thin} = 0.001, \Sigma_{thick} \Delta u_{thick} = 10$

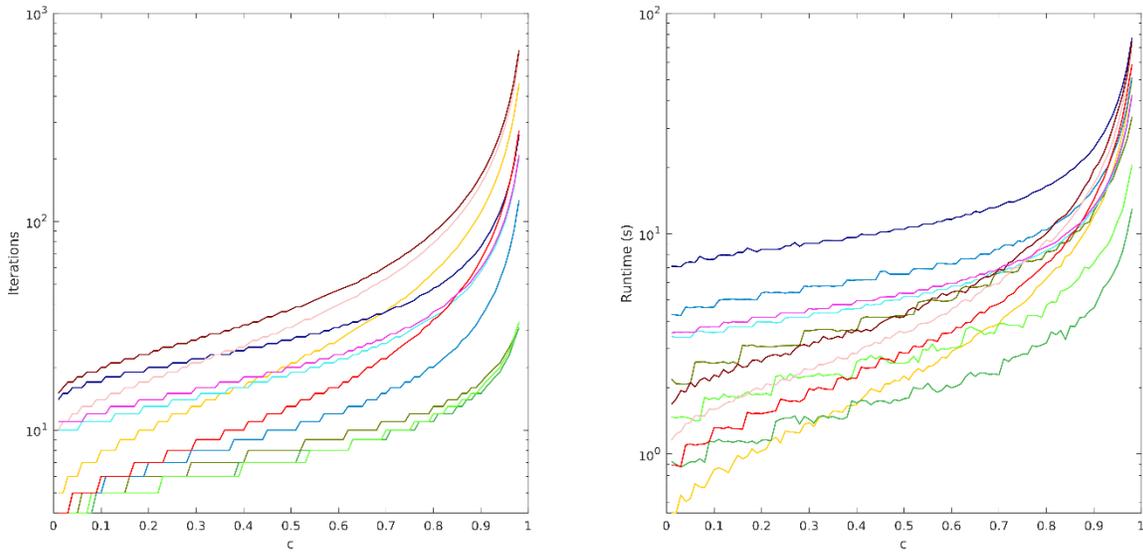


Figure A.6: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 1$

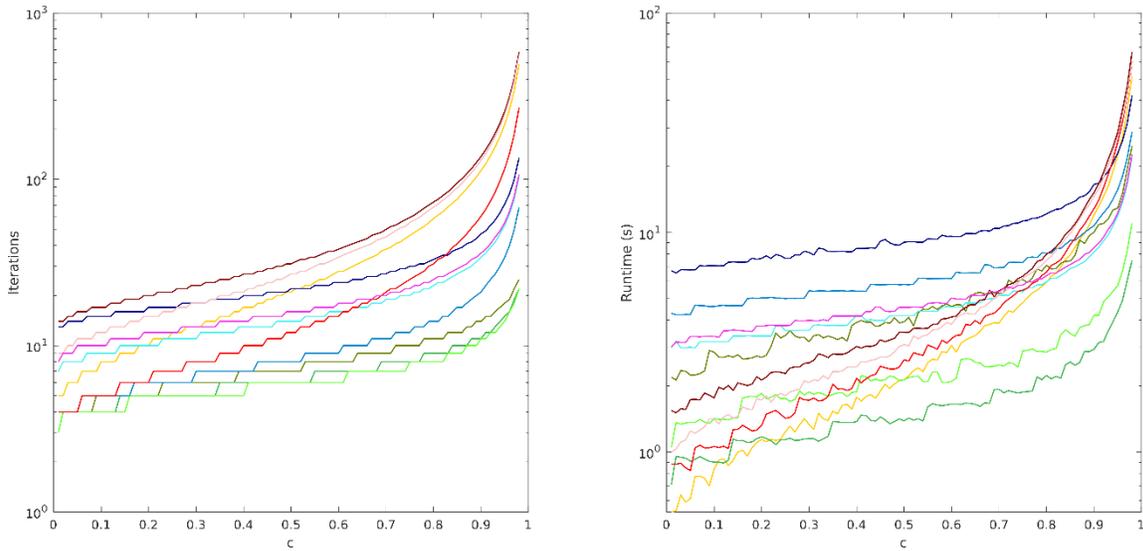


Figure A.7: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 2.5$

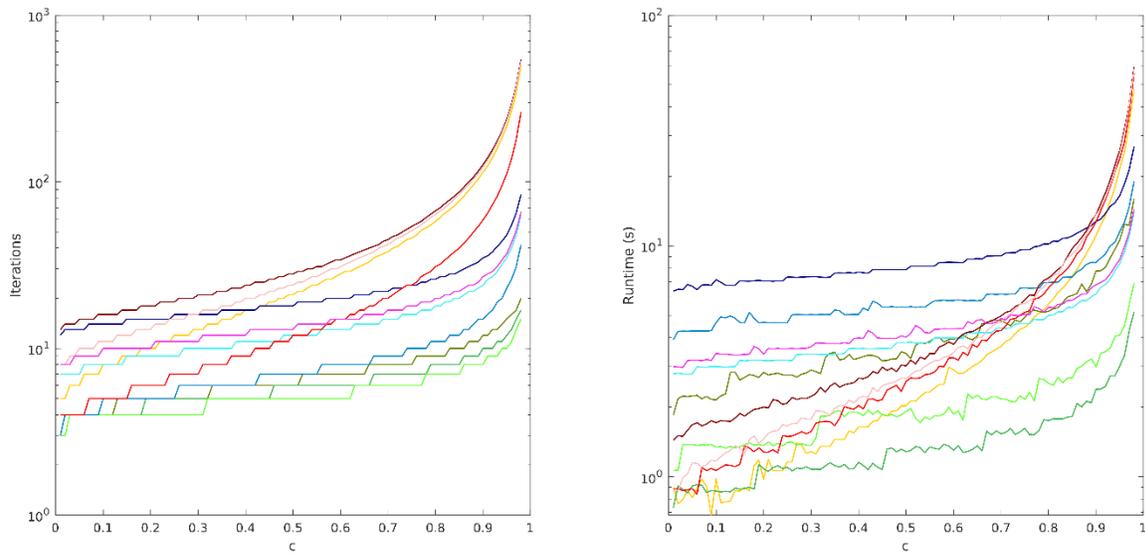


Figure A.8: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$

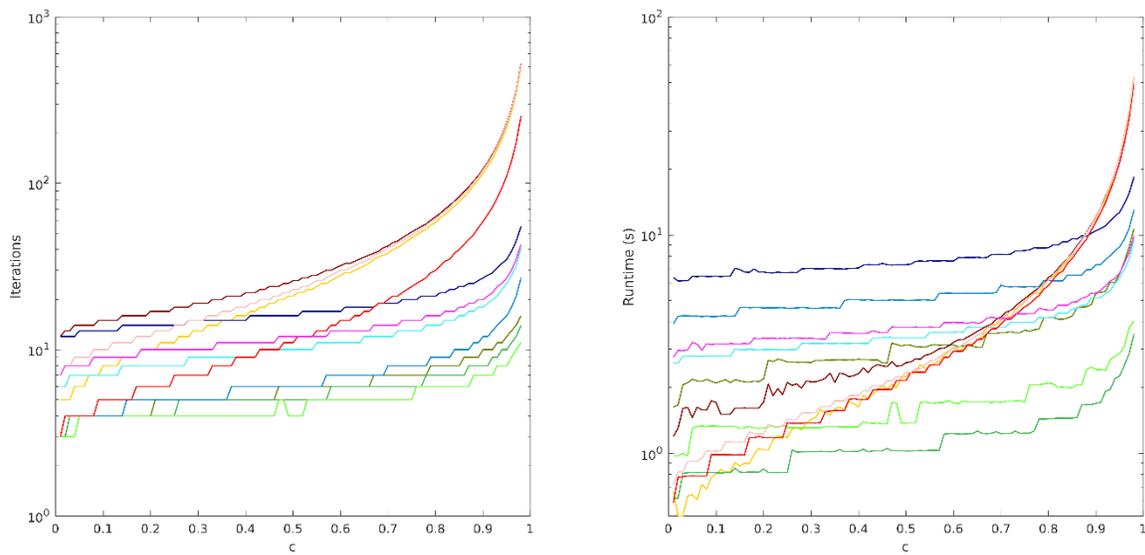


Figure A.9: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 10$

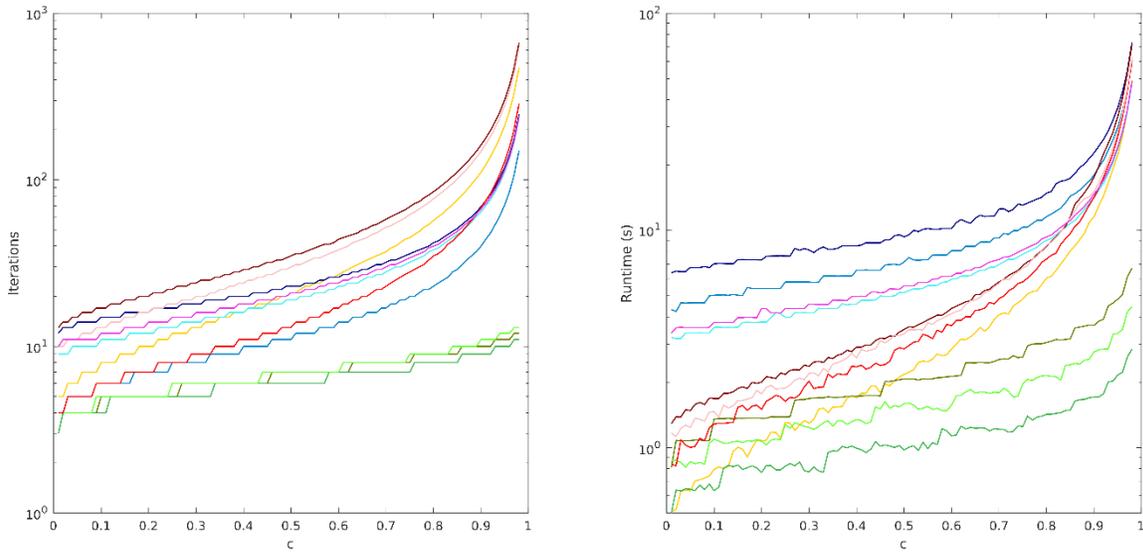


Figure A.10: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 1$

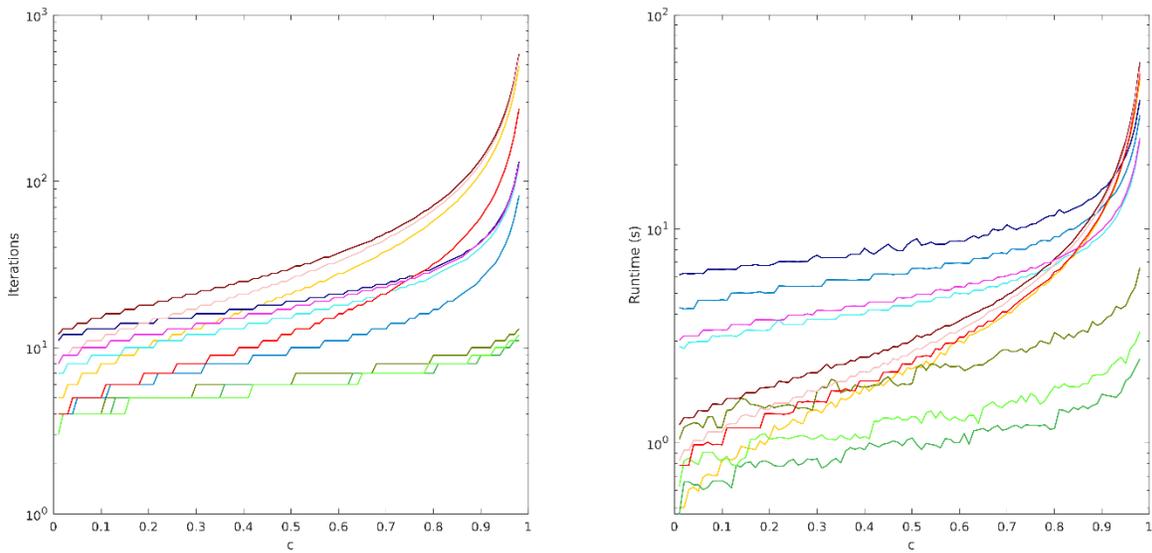


Figure A.11: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 2.5$

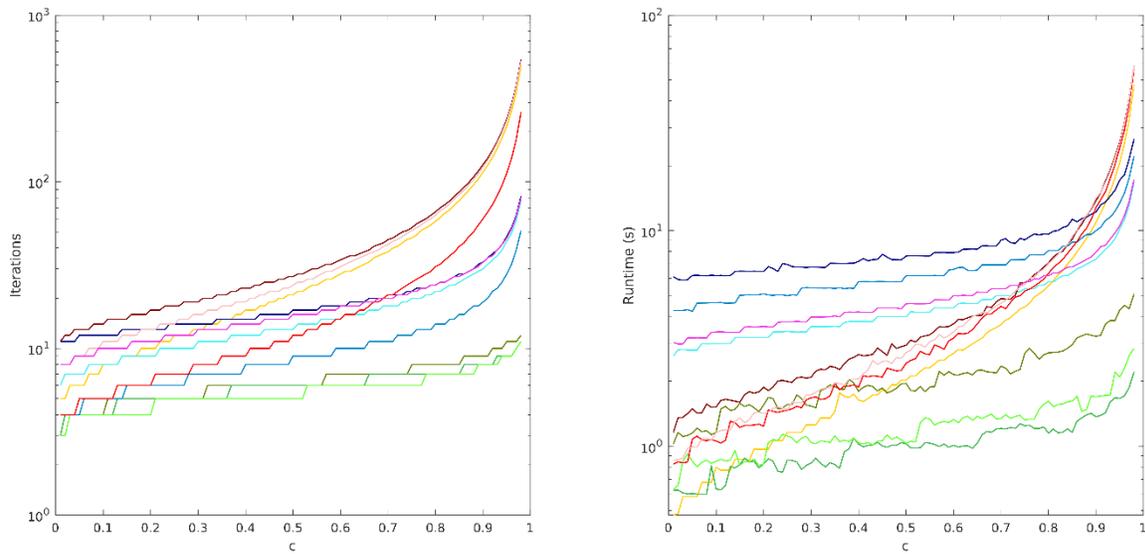


Figure A.12: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 5$

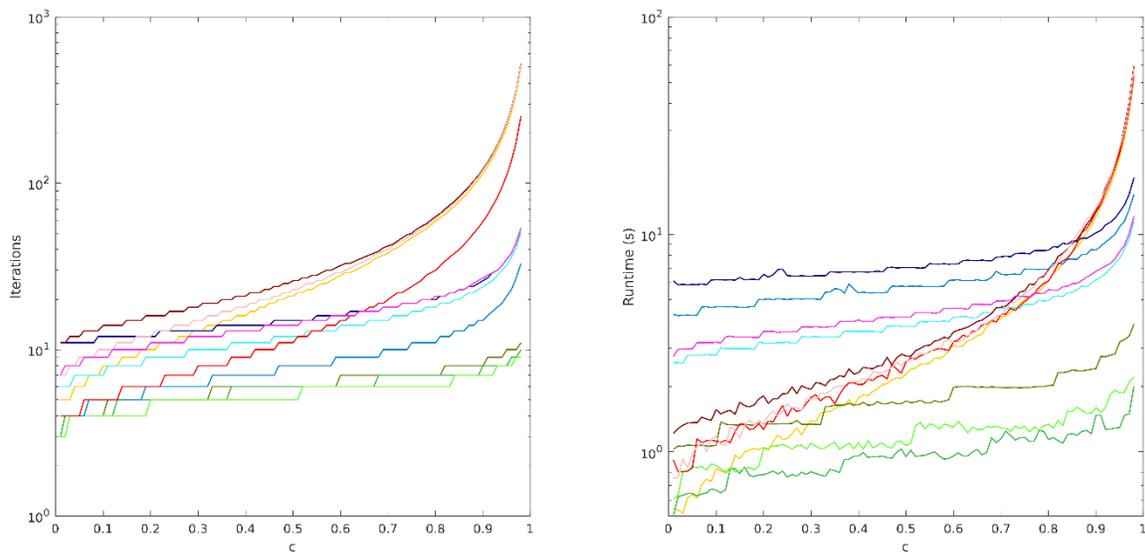


Figure A.13: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 10$

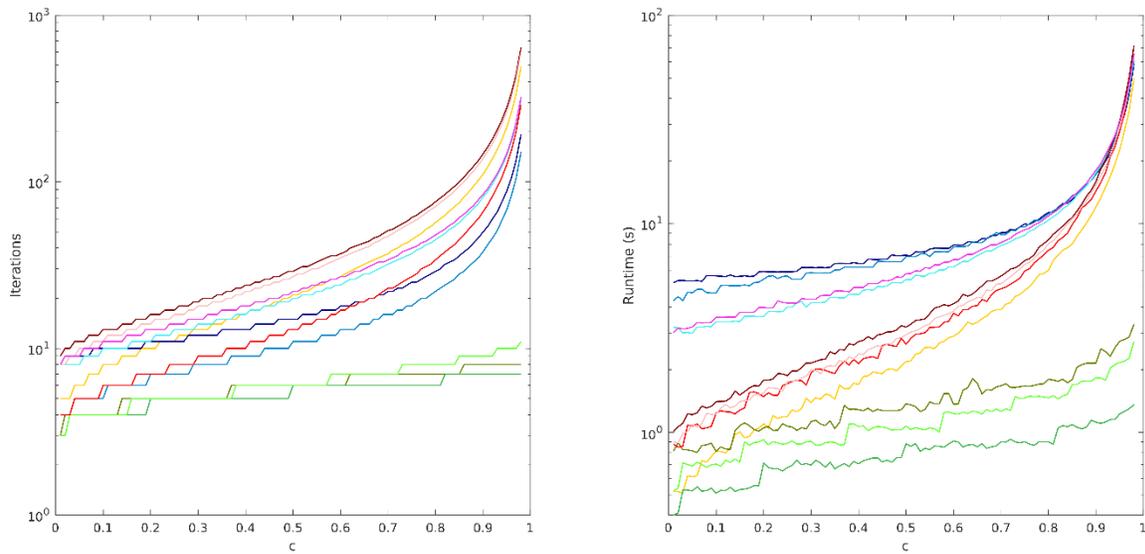


Figure A.14: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 1$

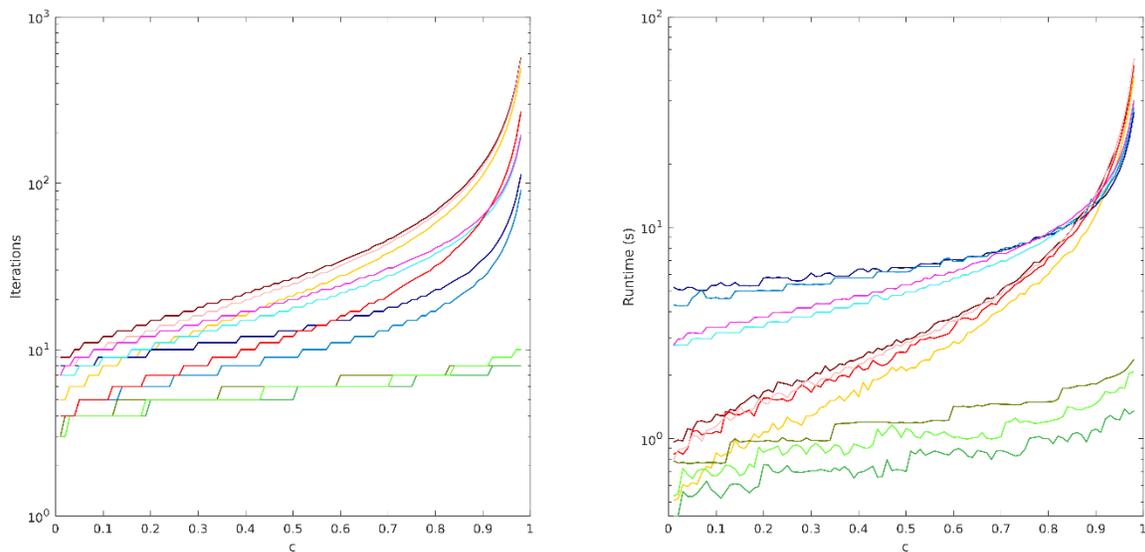


Figure A.15: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 2.5$

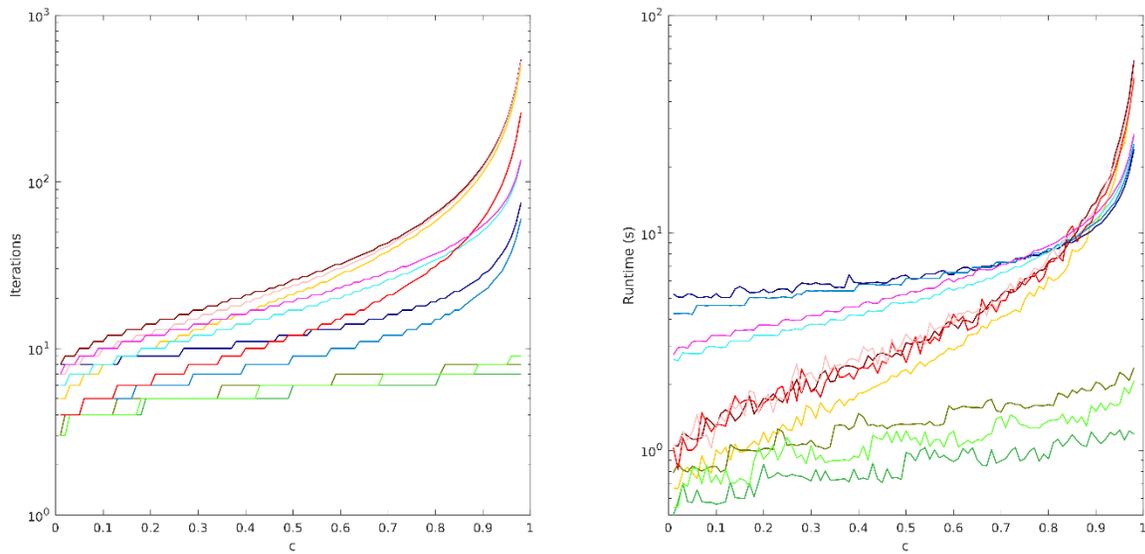


Figure A.16: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 5$

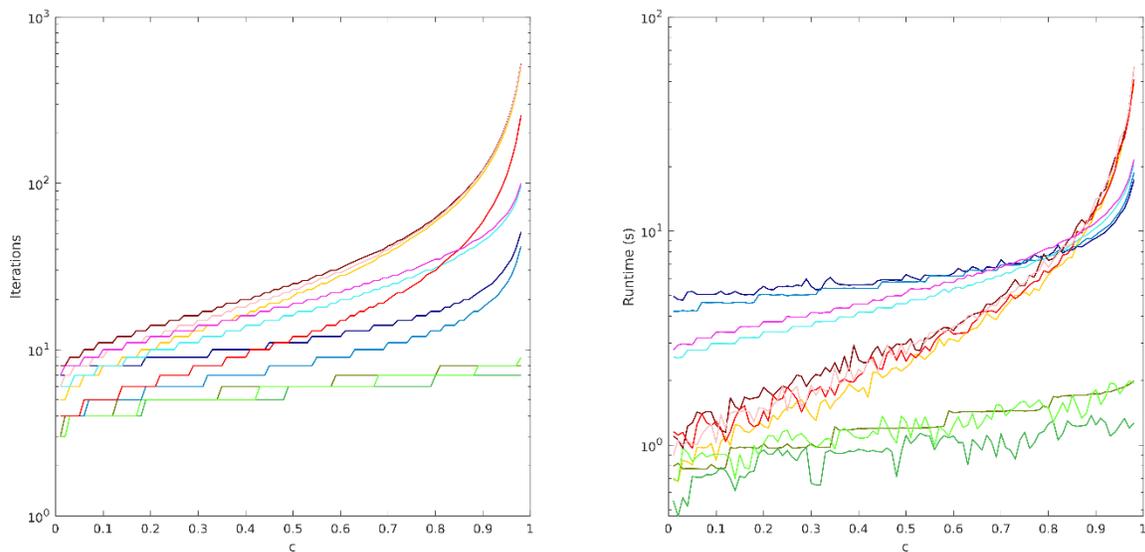


Figure A.17: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 10$

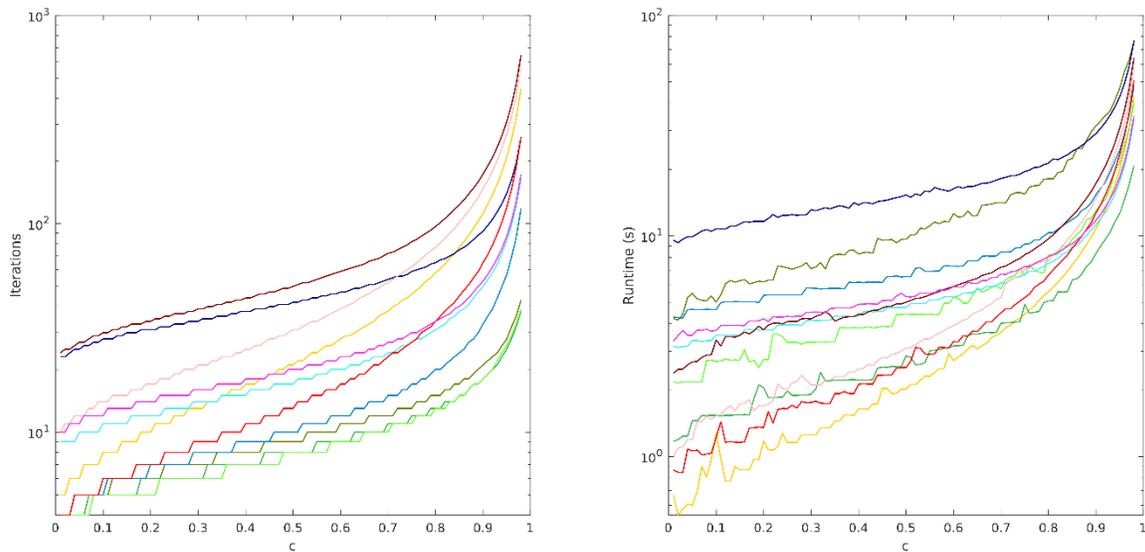


Figure A.18: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 1$

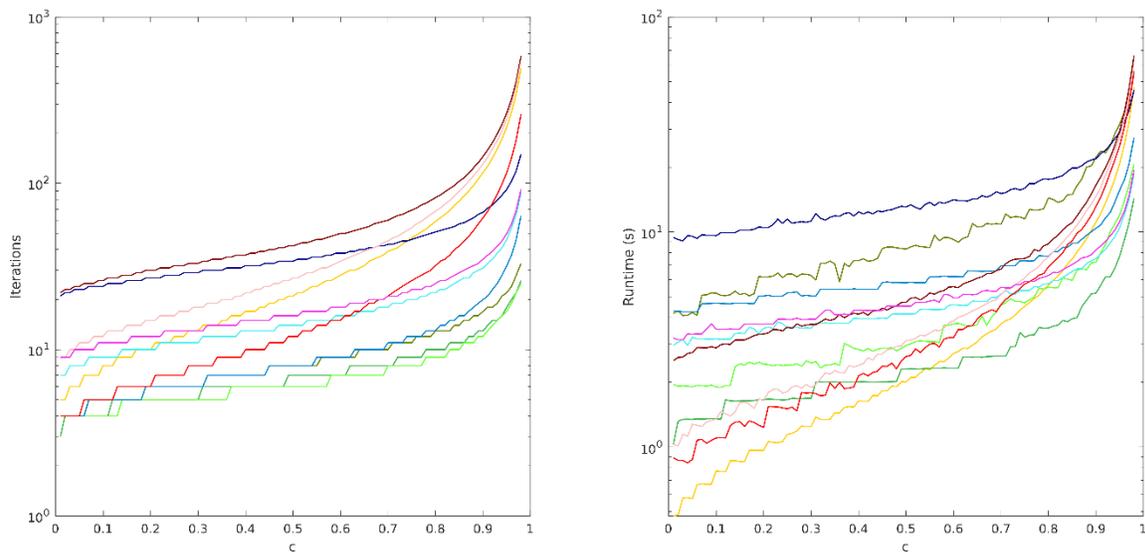


Figure A.19: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 2.5$

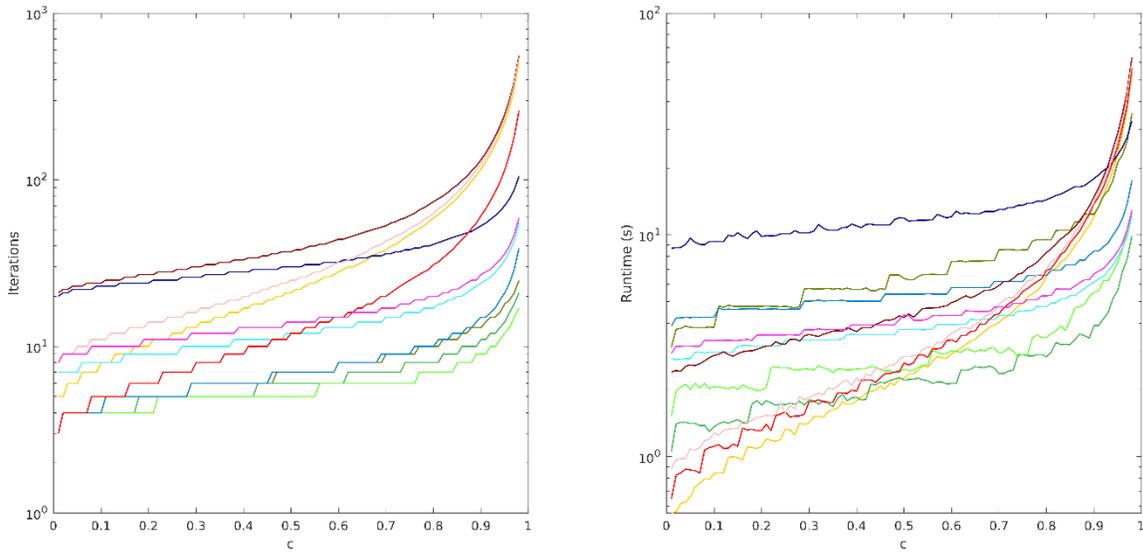


Figure A.20: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin} \Delta u_{thin} = 0.001, \Sigma_{thick} \Delta u_{thick} = 5$

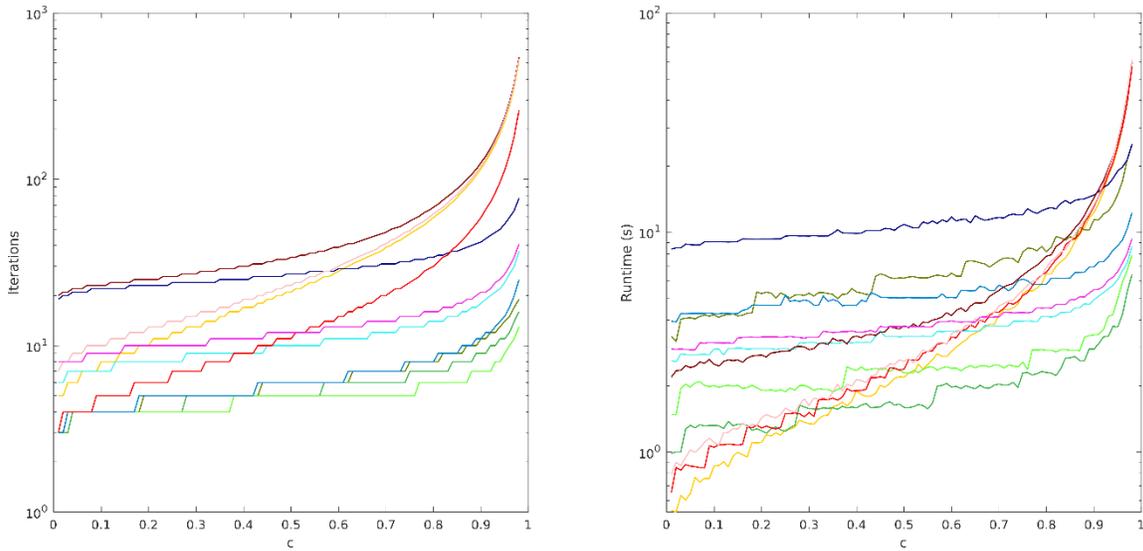


Figure A.21: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin} \Delta u_{thin} = 0.001, \Sigma_{thick} \Delta u_{thick} = 10$

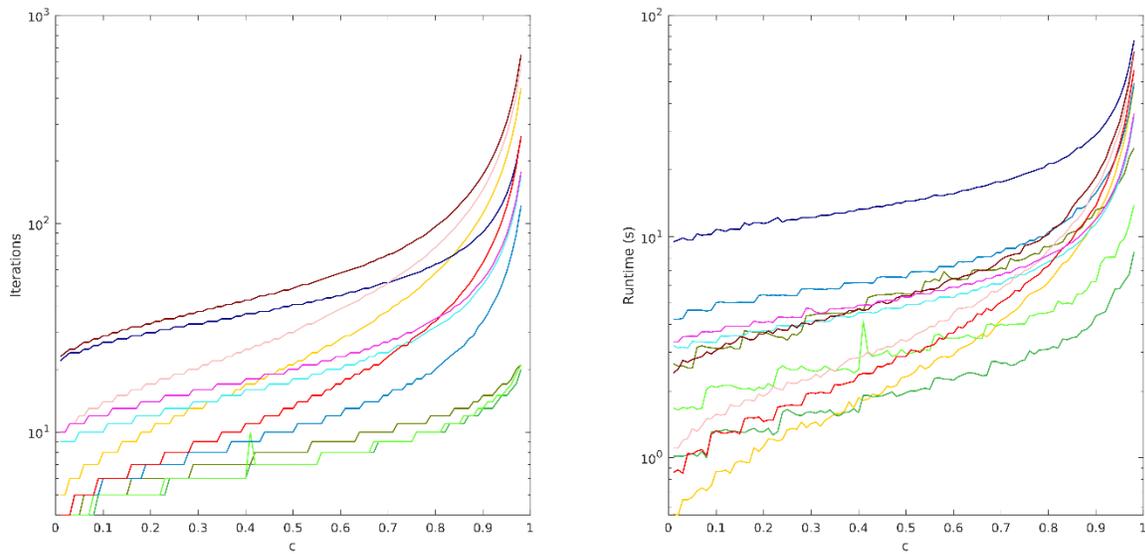


Figure A.22: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 1$

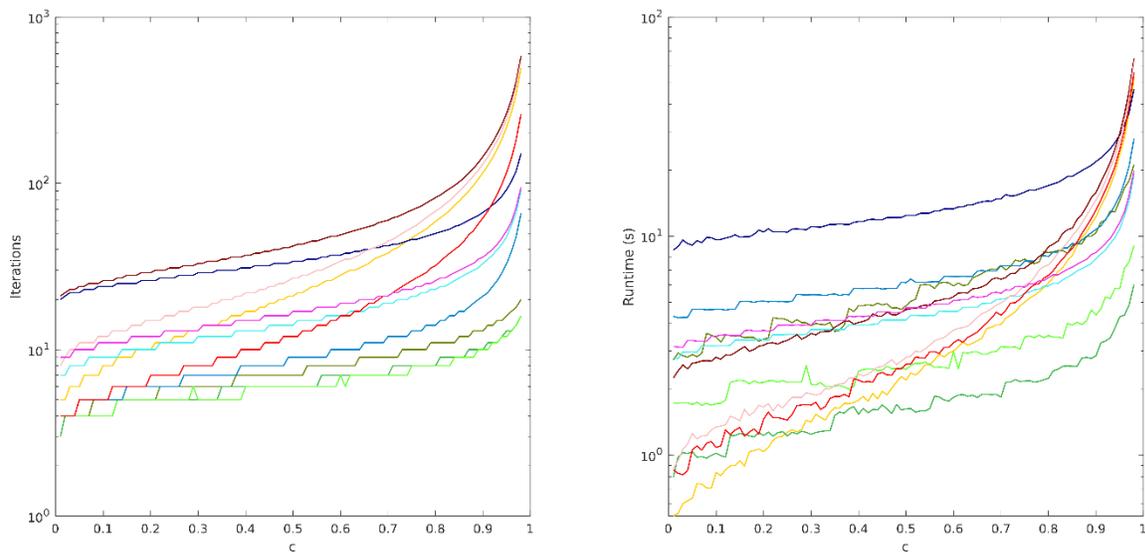


Figure A.23: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 2.5$

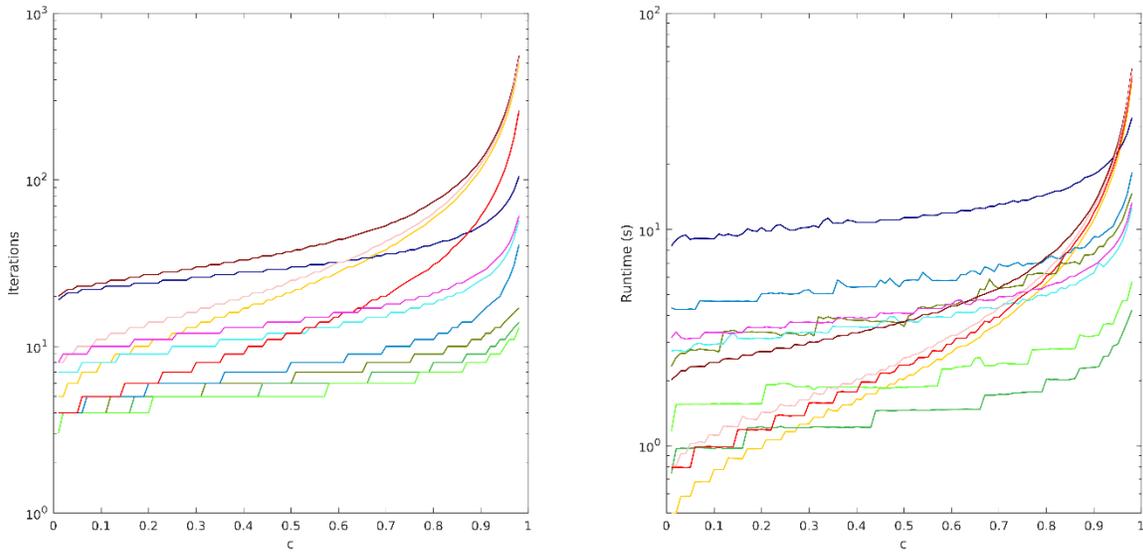


Figure A.24: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$

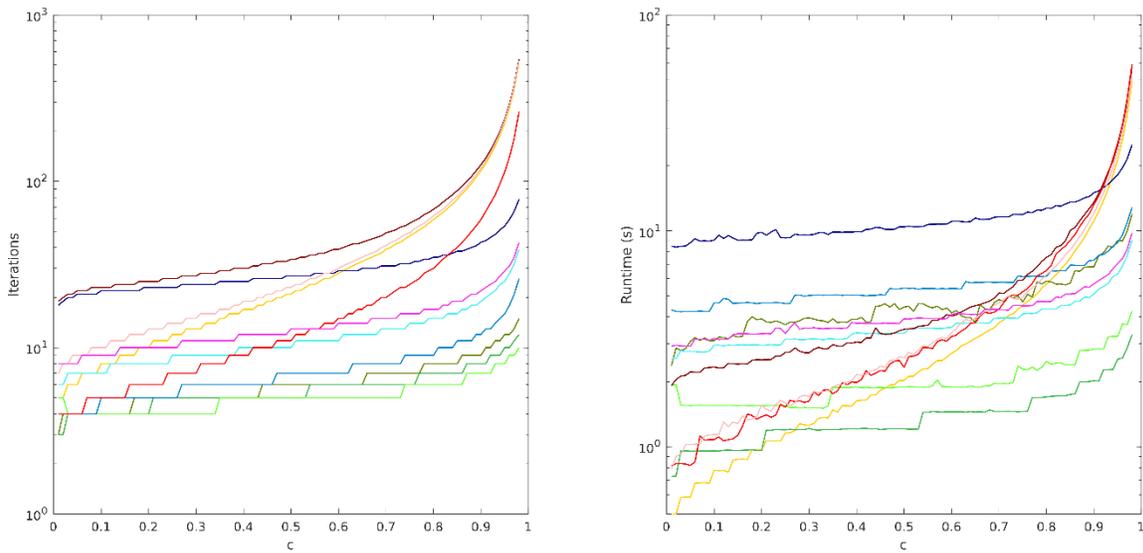


Figure A.25: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 10$

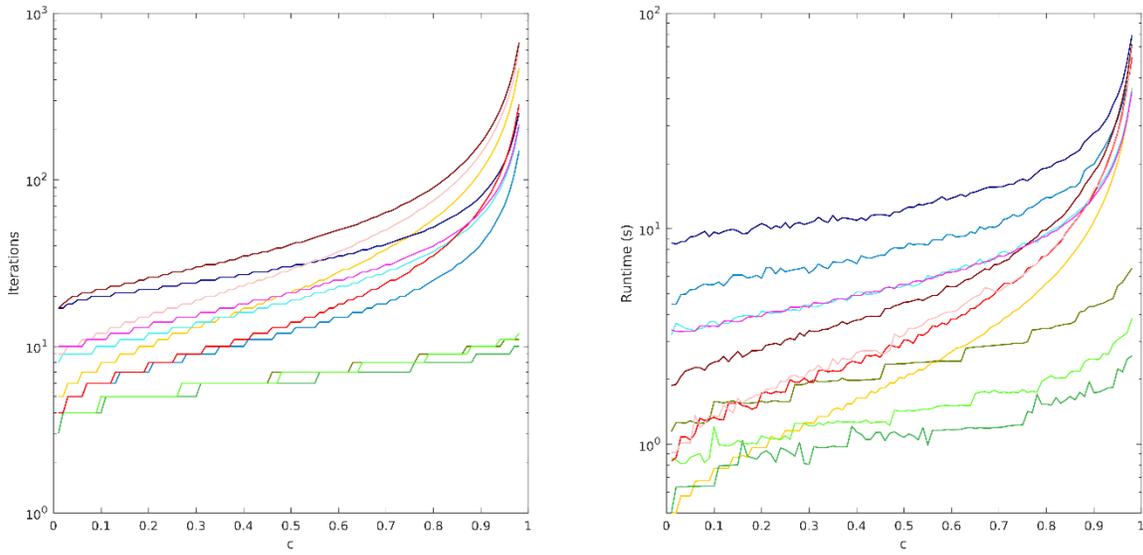


Figure A.26: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 1$

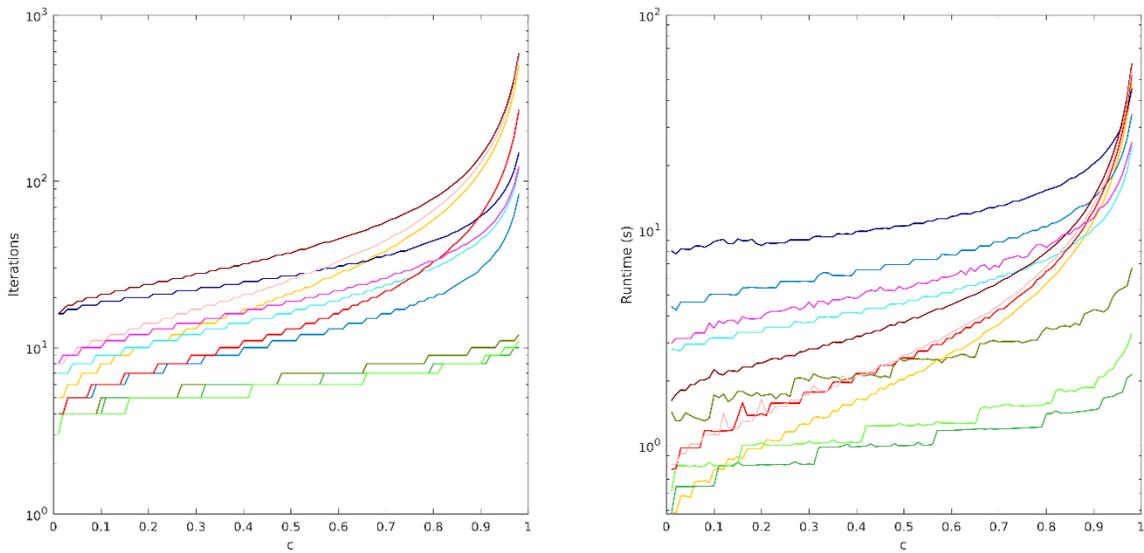


Figure A.27: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 2.5$

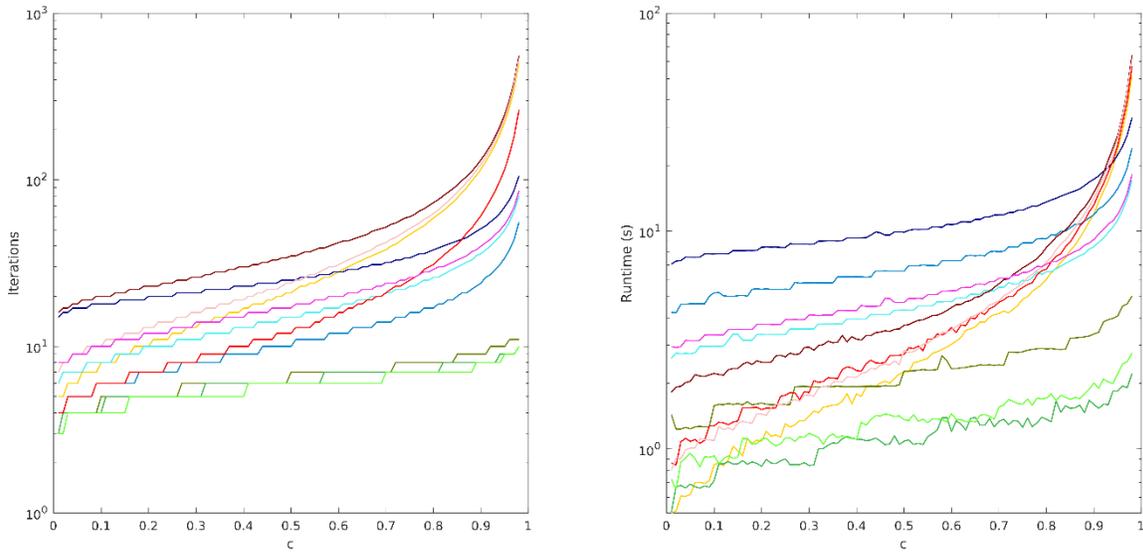


Figure A.28: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 5$

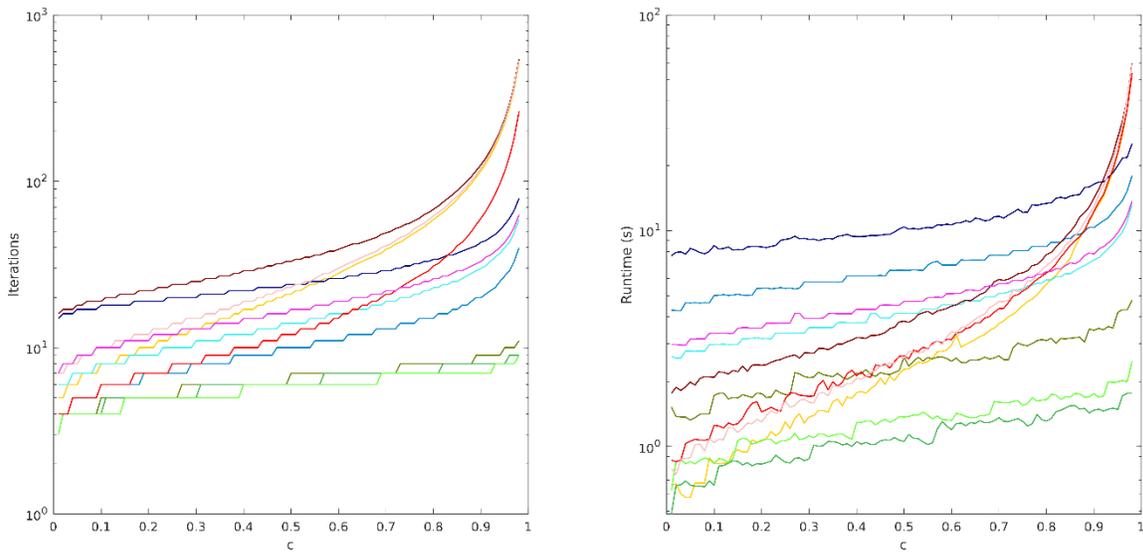


Figure A.29: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 10$

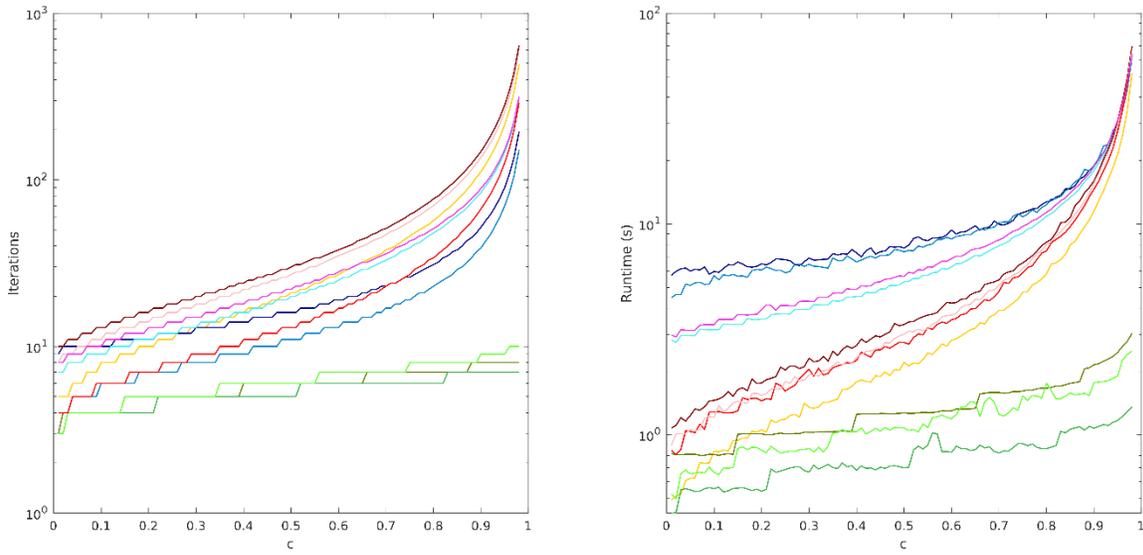


Figure A.30: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 1$

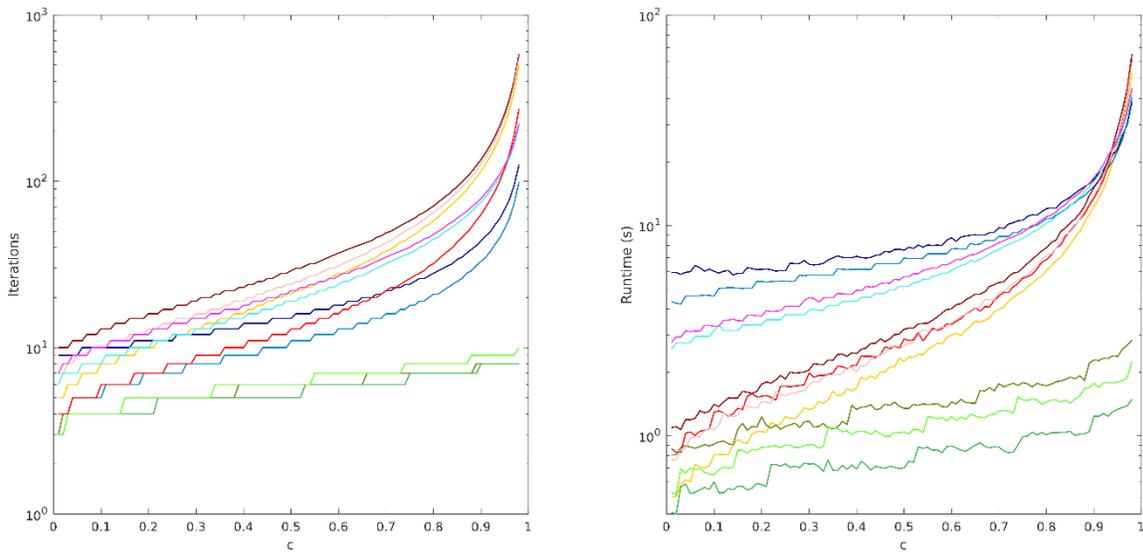


Figure A.31: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 2.5$

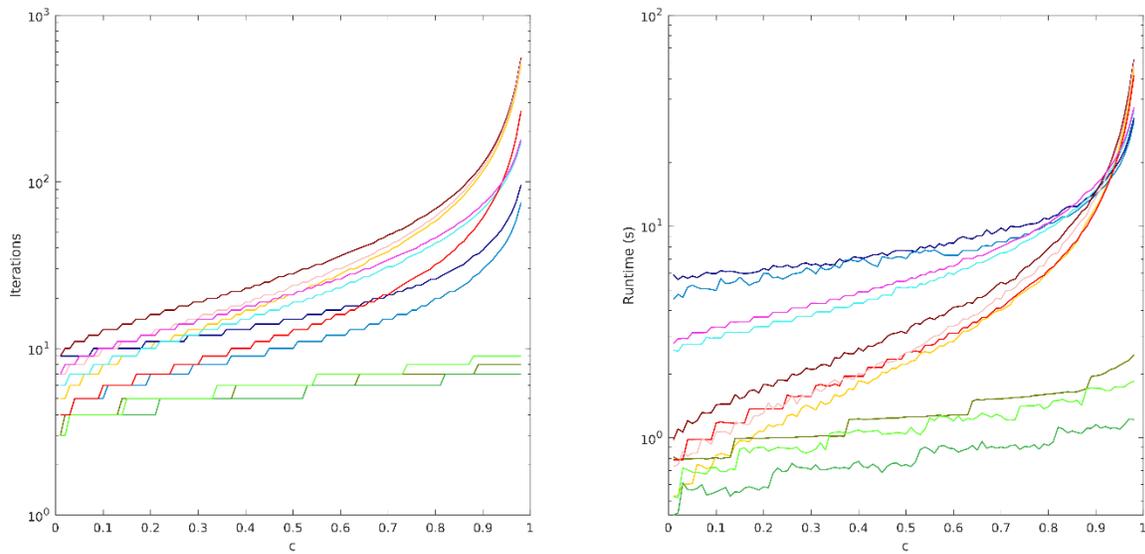


Figure A.32: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 5$

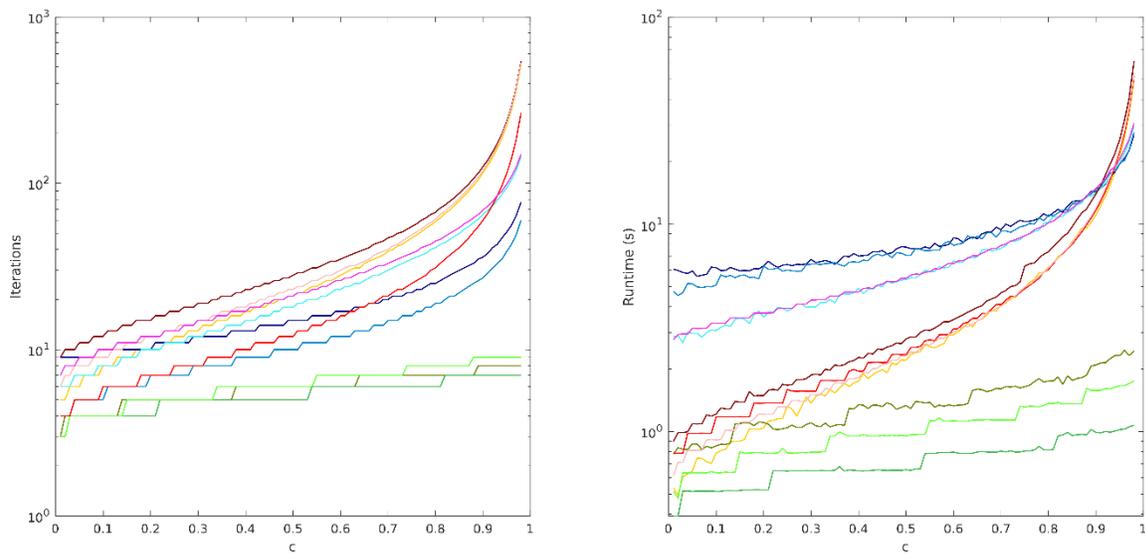


Figure A.33: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 10$

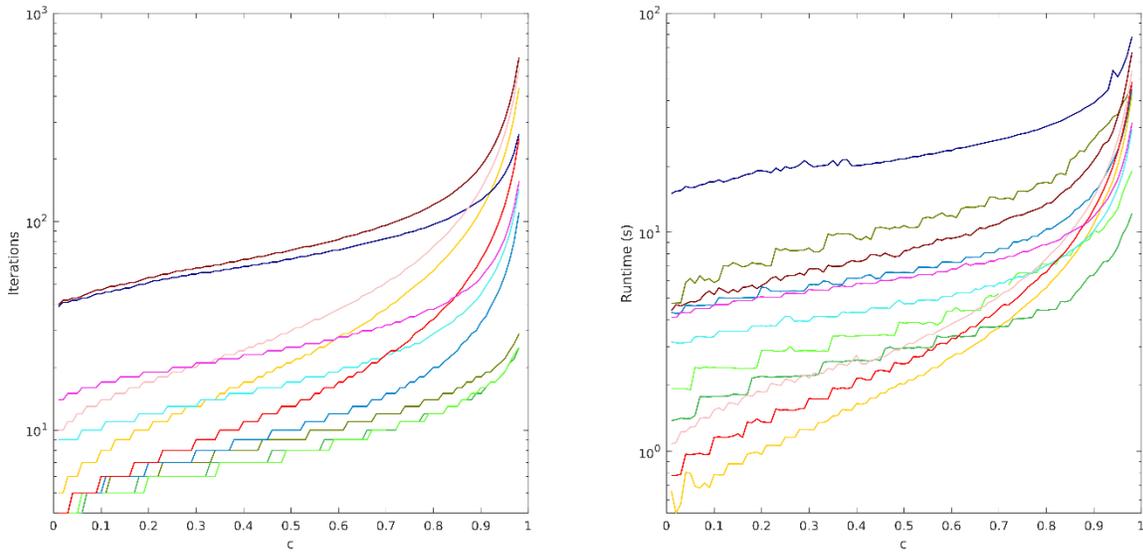


Figure A.34: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

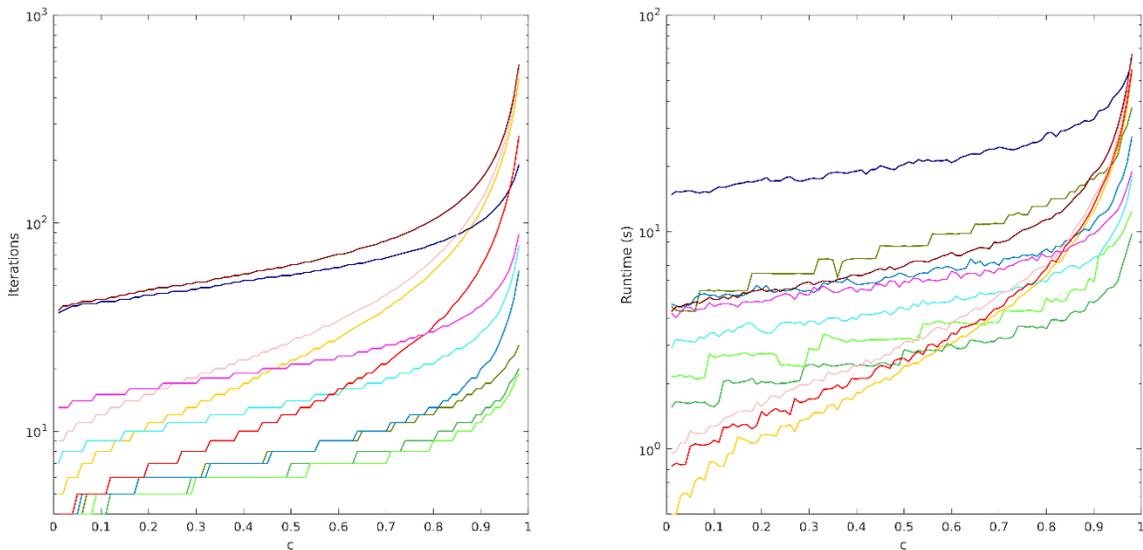


Figure A.35: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

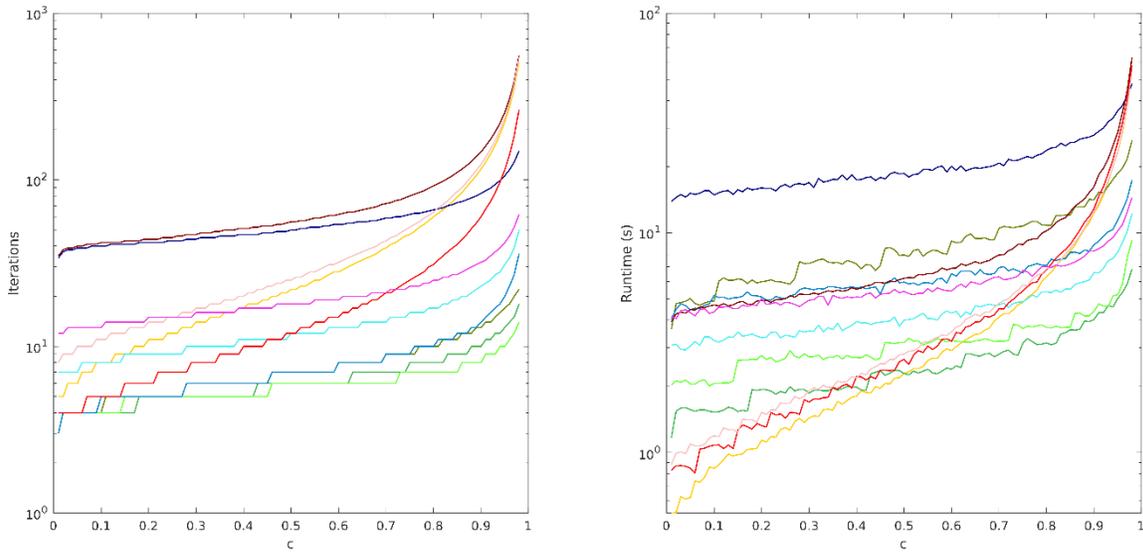


Figure A.36: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

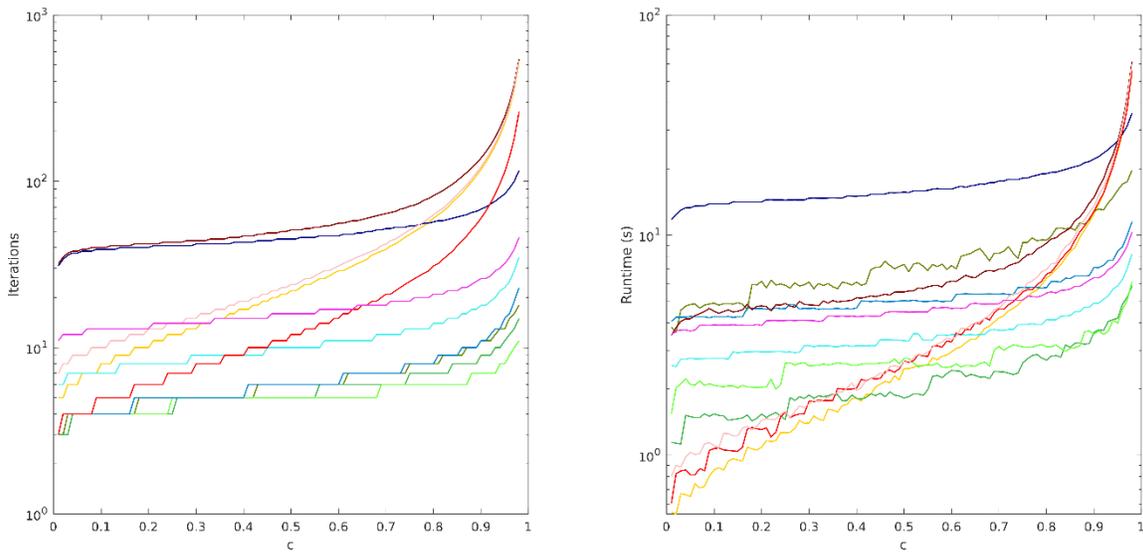


Figure A.37: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 10$

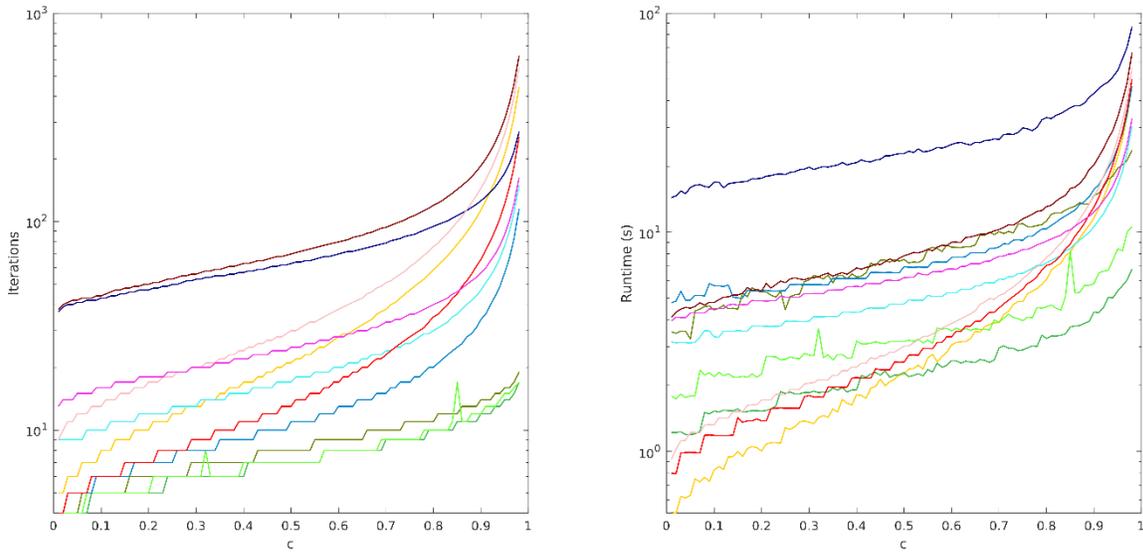


Figure A.38: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 1$

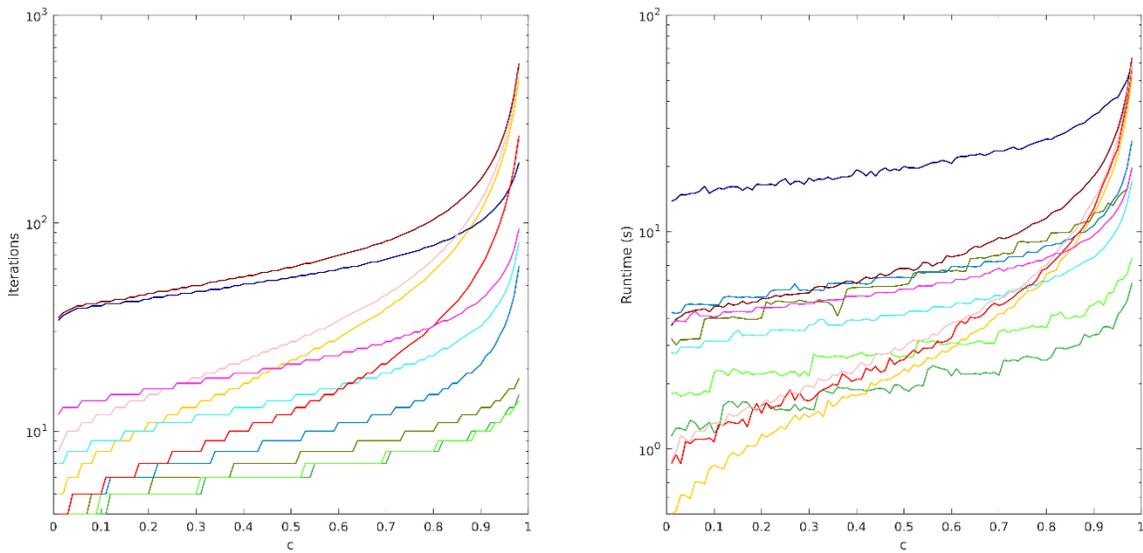


Figure A.39: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 2.5$

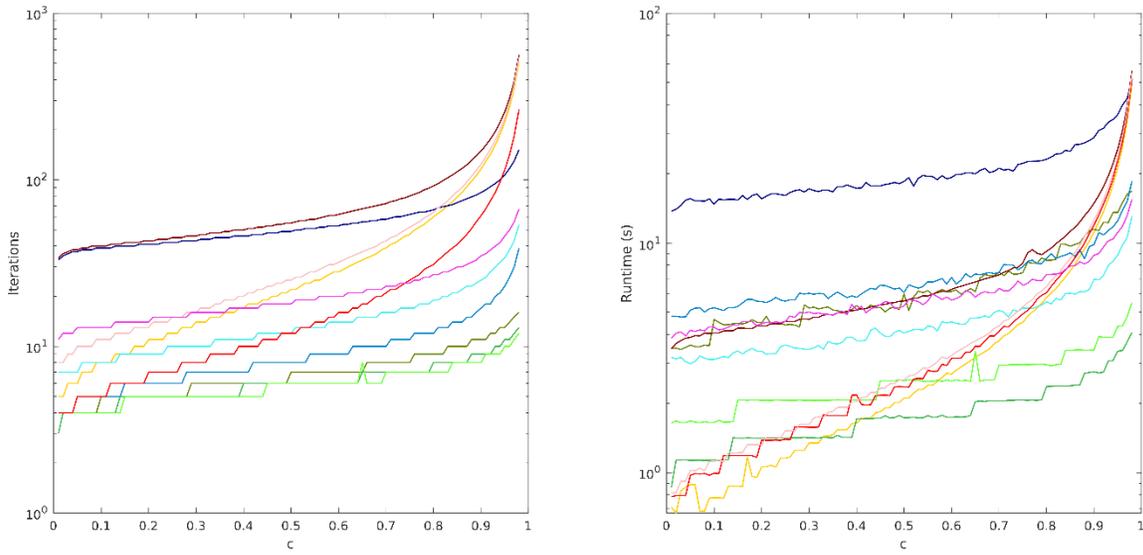


Figure A.40: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$

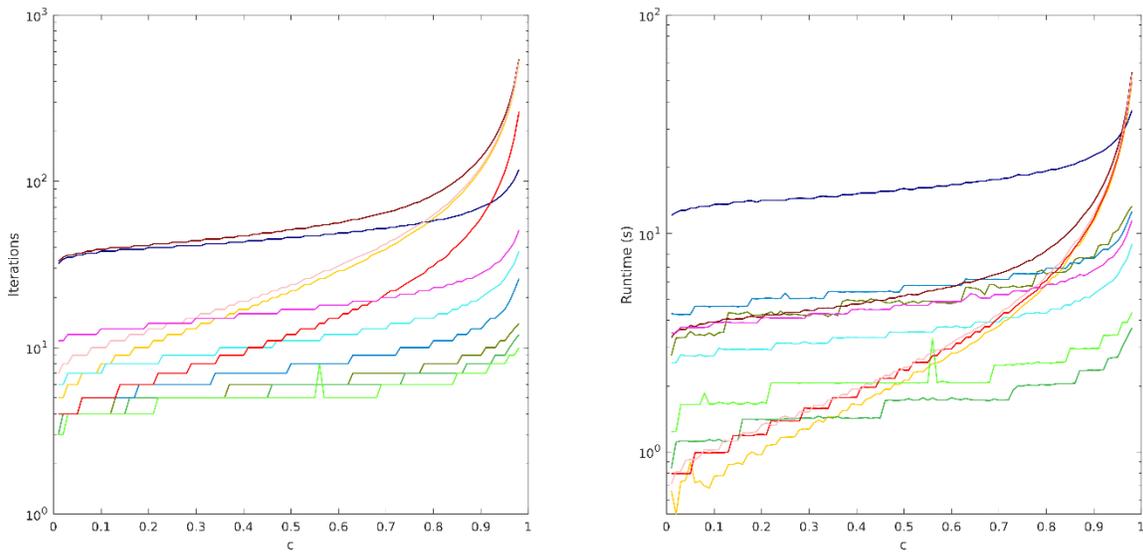


Figure A.41: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 10$

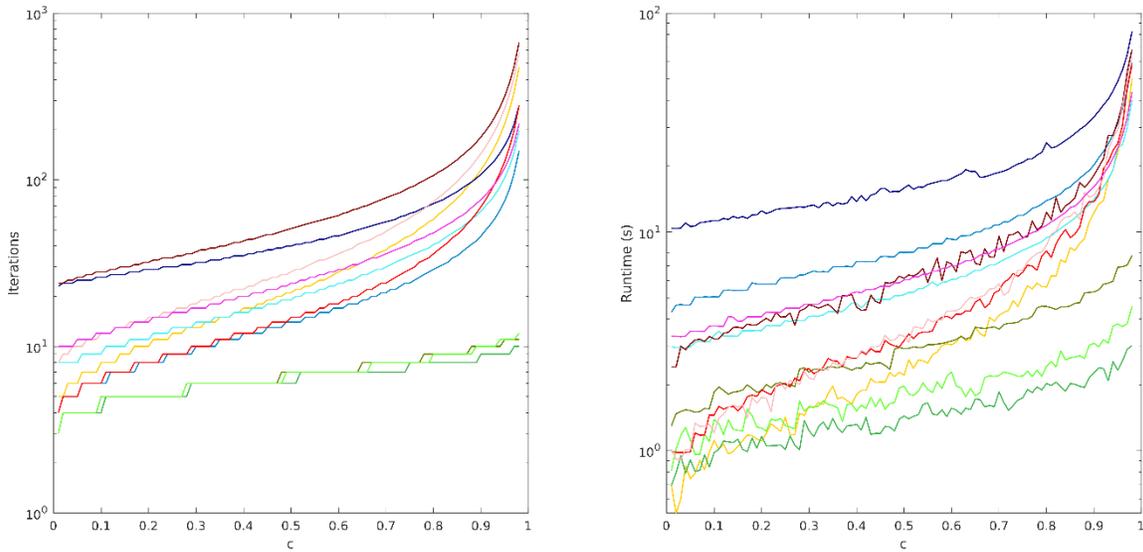


Figure A.42: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

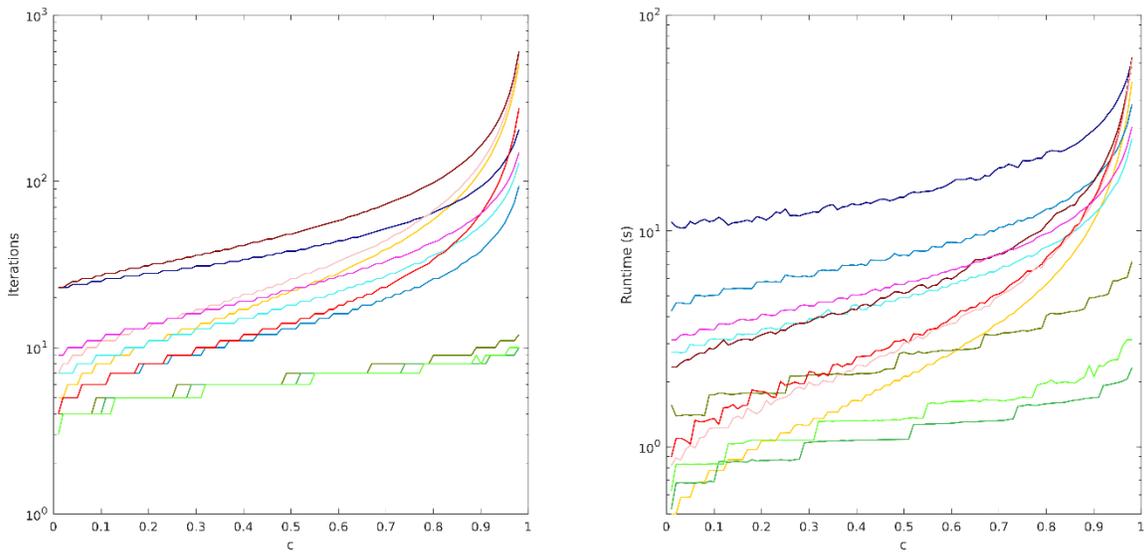


Figure A.43: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

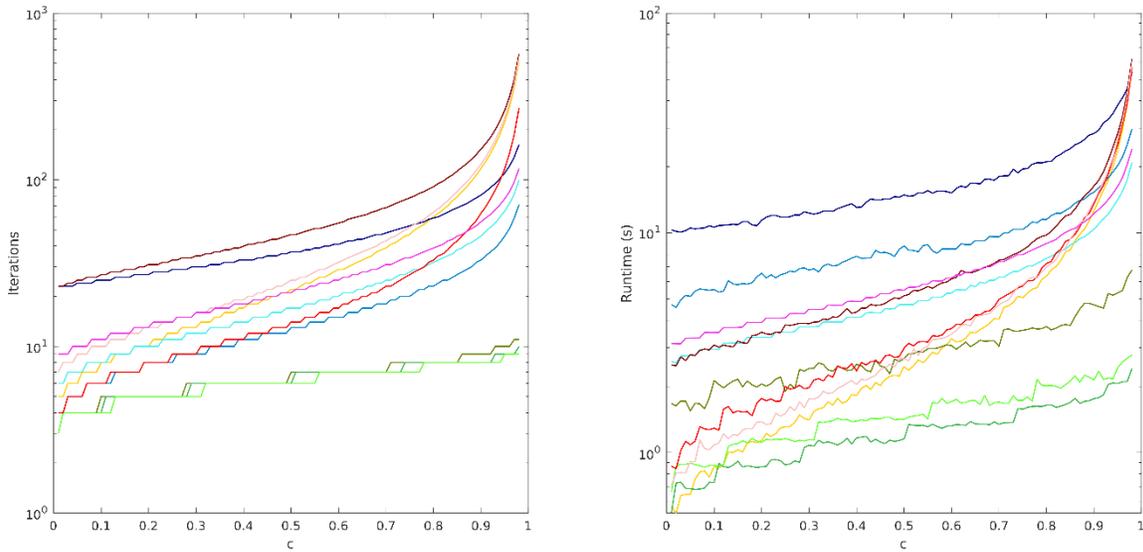


Figure A.44: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 5$

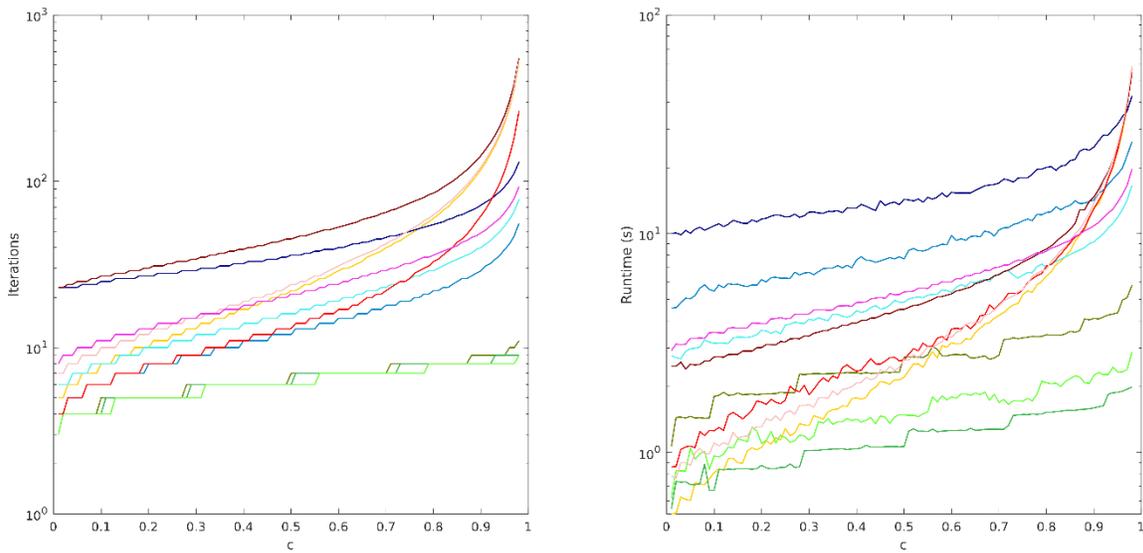


Figure A.45: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 10$

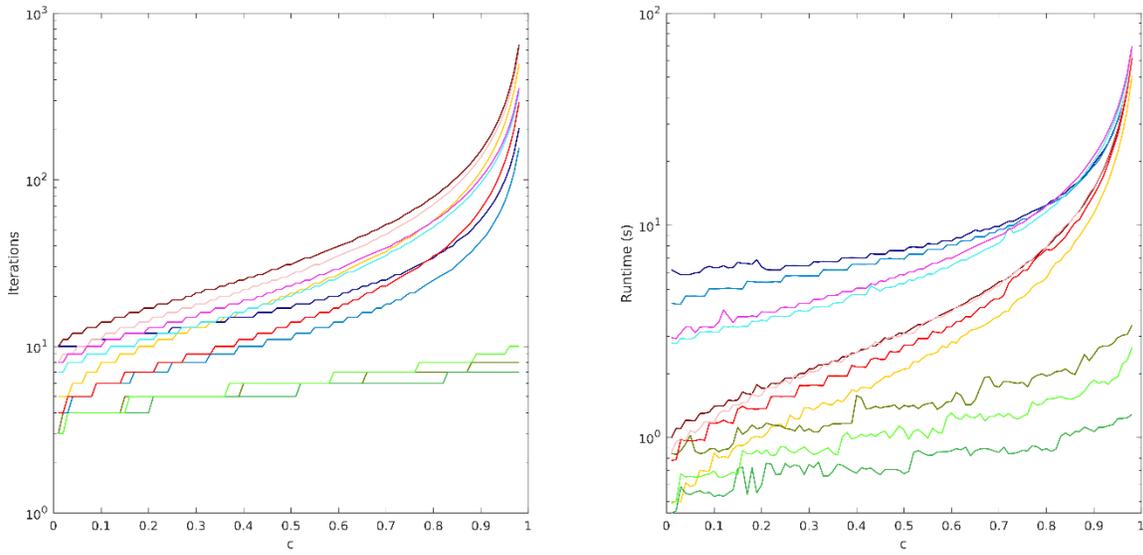


Figure A.46: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

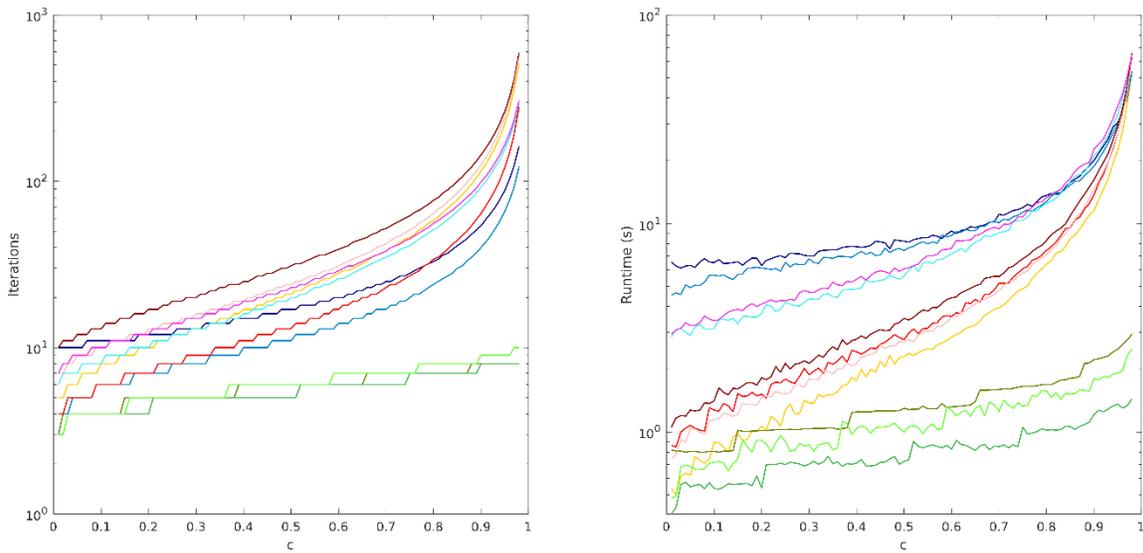


Figure A.47: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

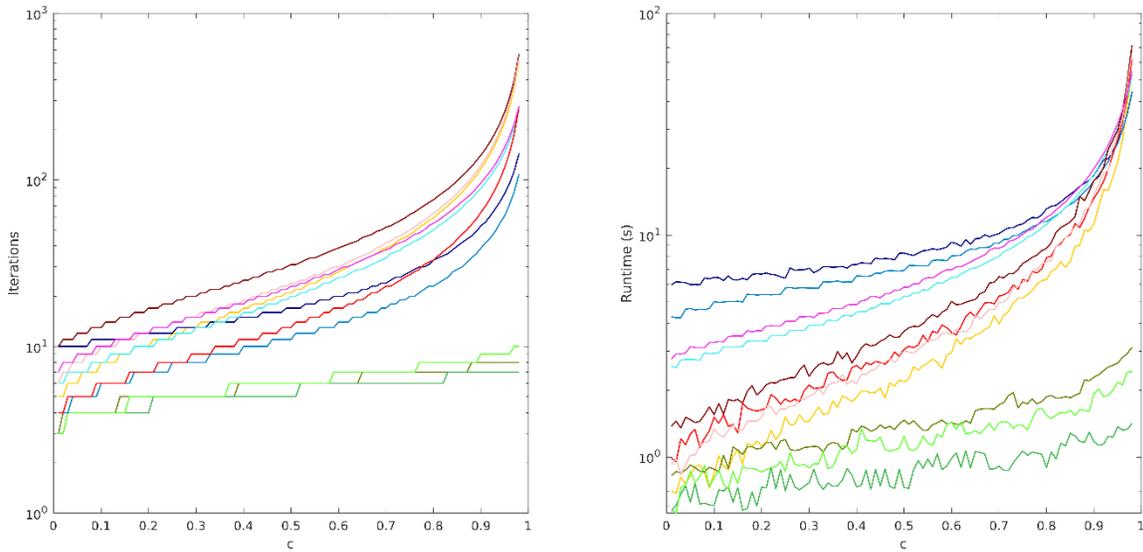


Figure A.48: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 5$

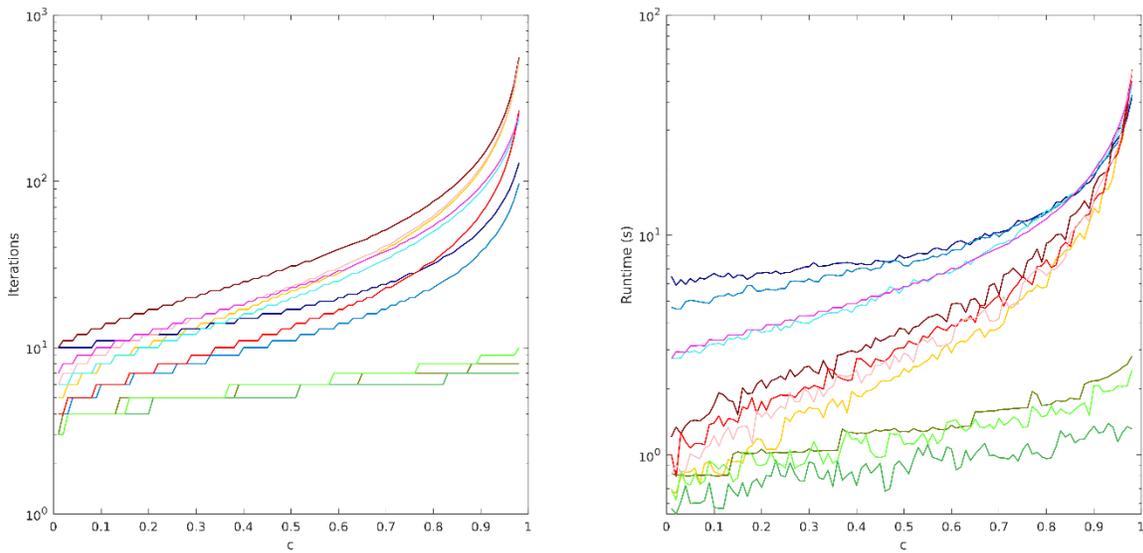


Figure A.49: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 10$

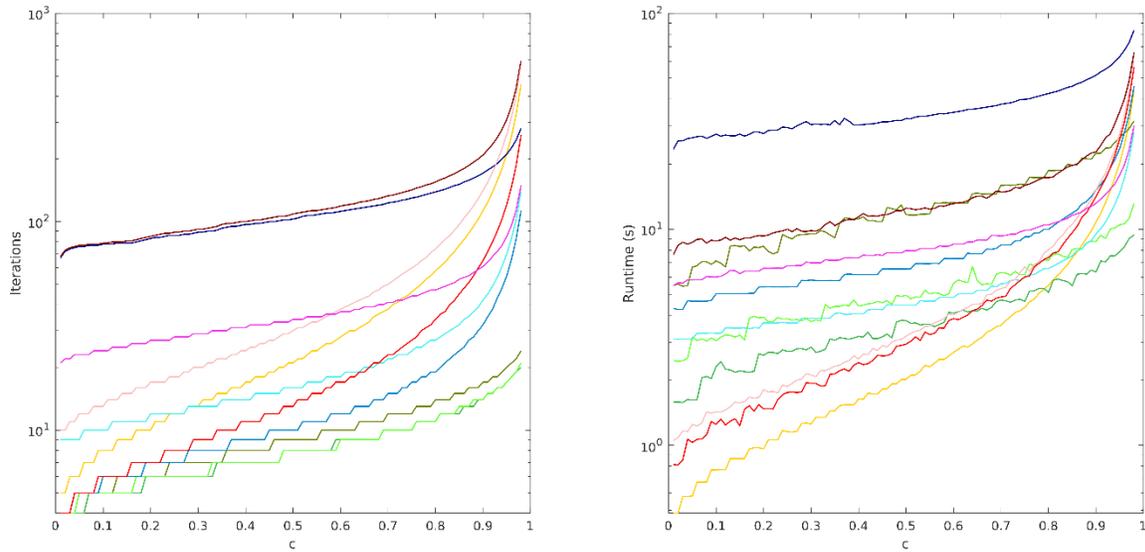


Figure A.50: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

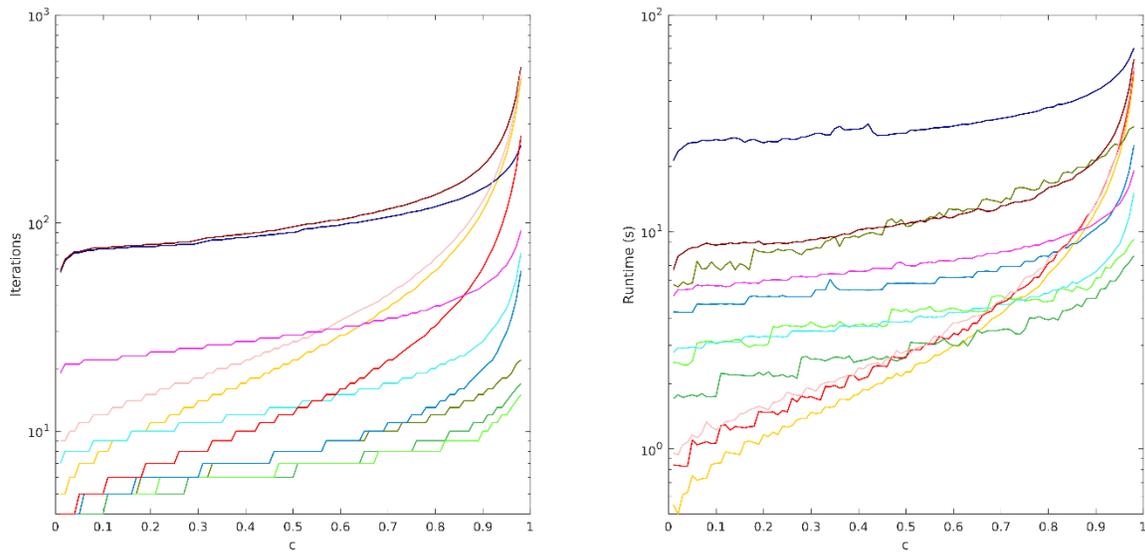


Figure A.51: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

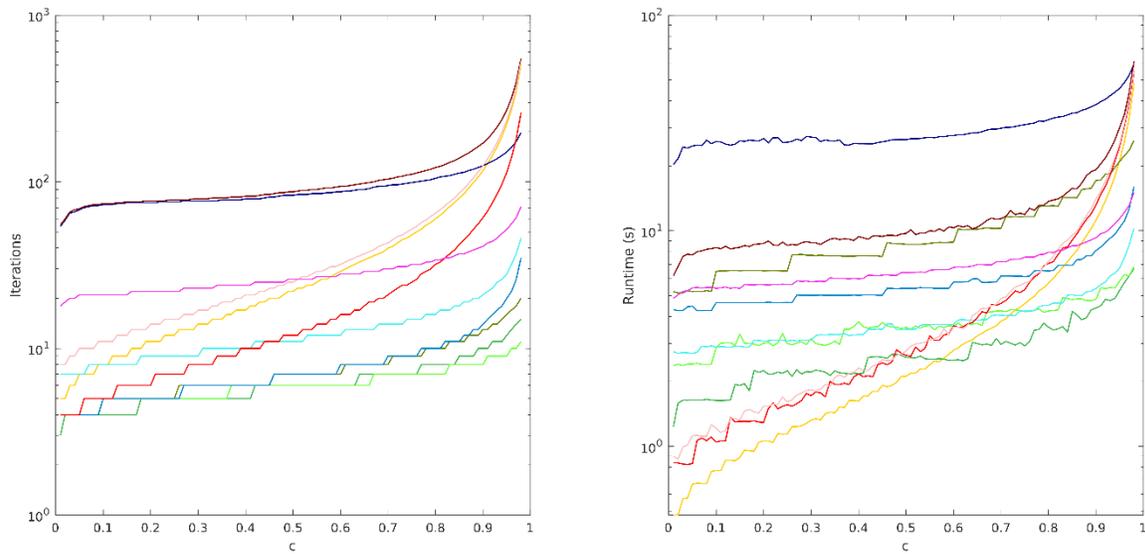


Figure A.52: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 5$

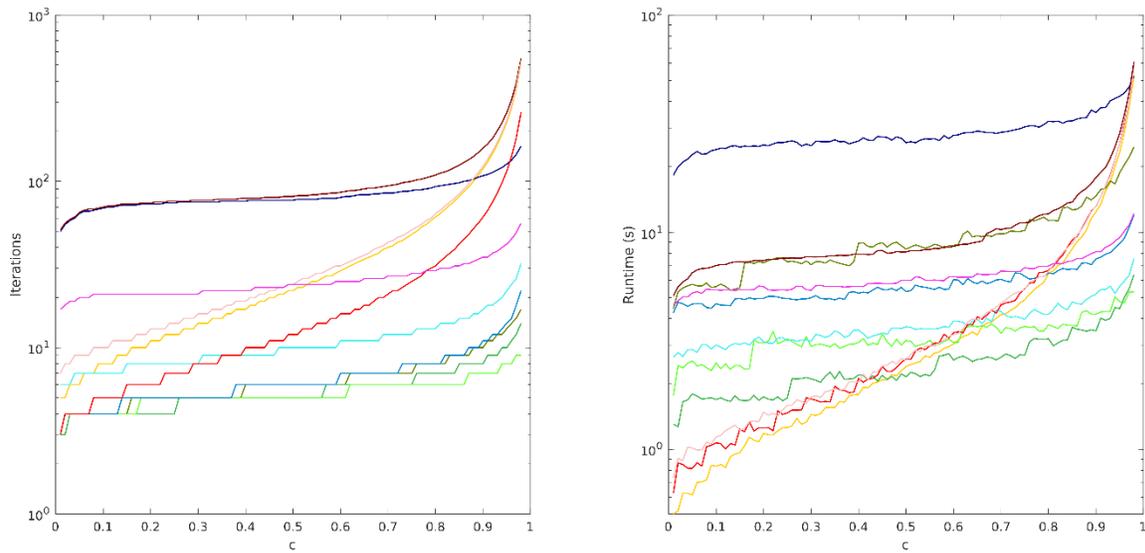


Figure A.53: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 10$

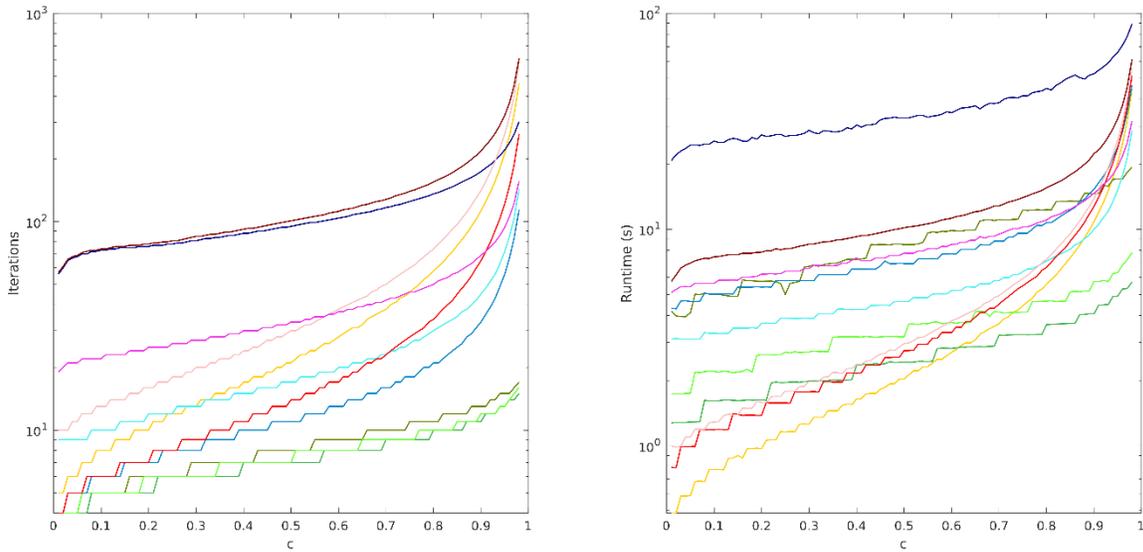


Figure A.54: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 1$

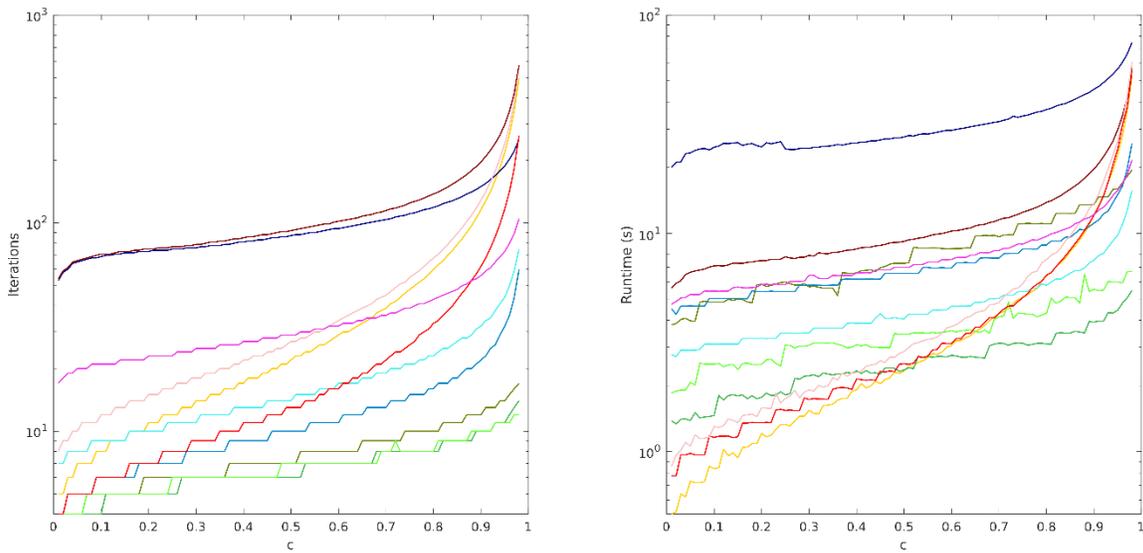


Figure A.55: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 2.5$

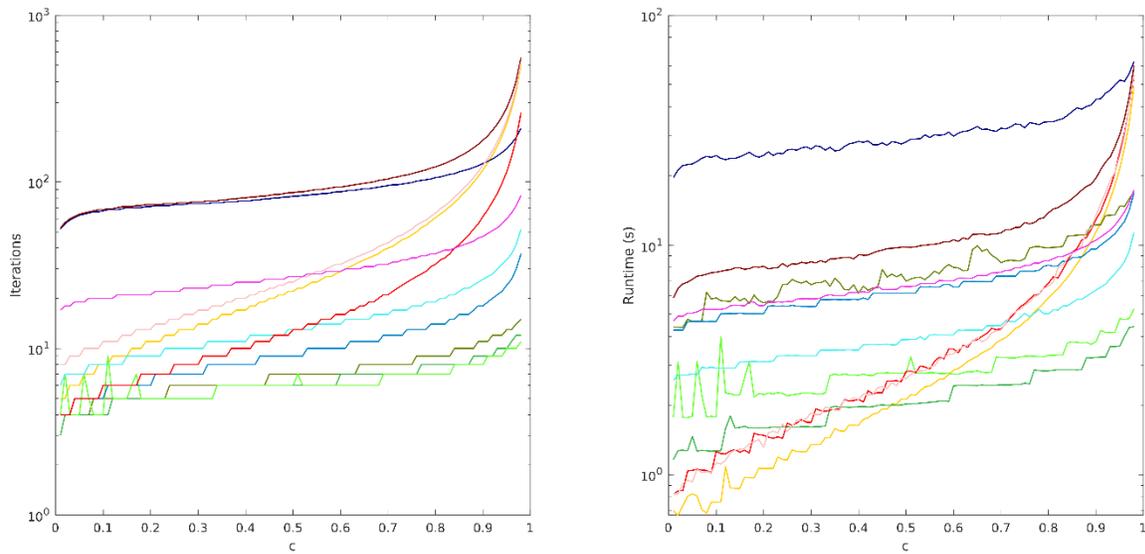


Figure A.56: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$

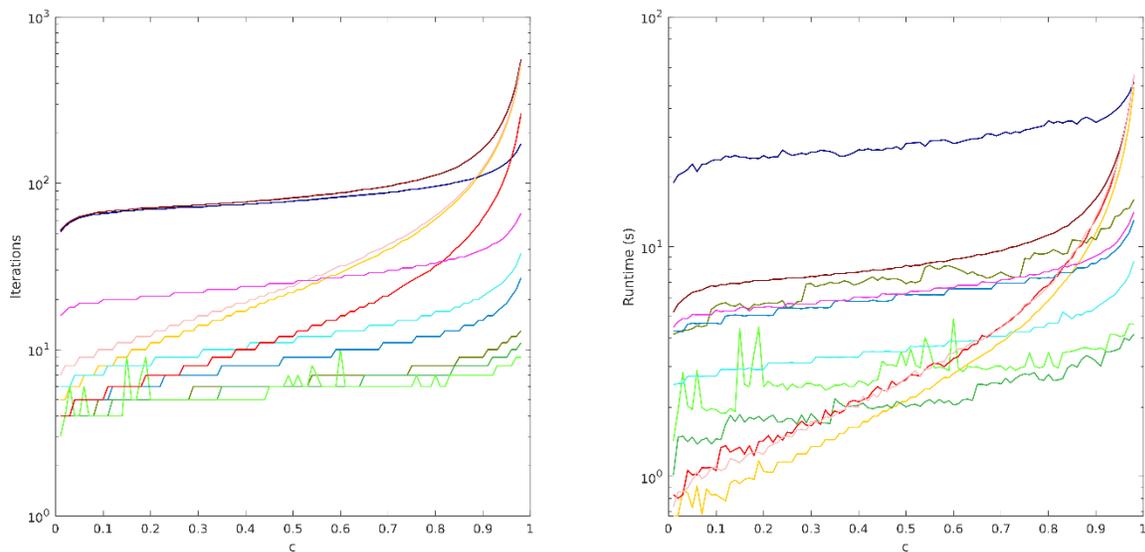


Figure A.57: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 10$

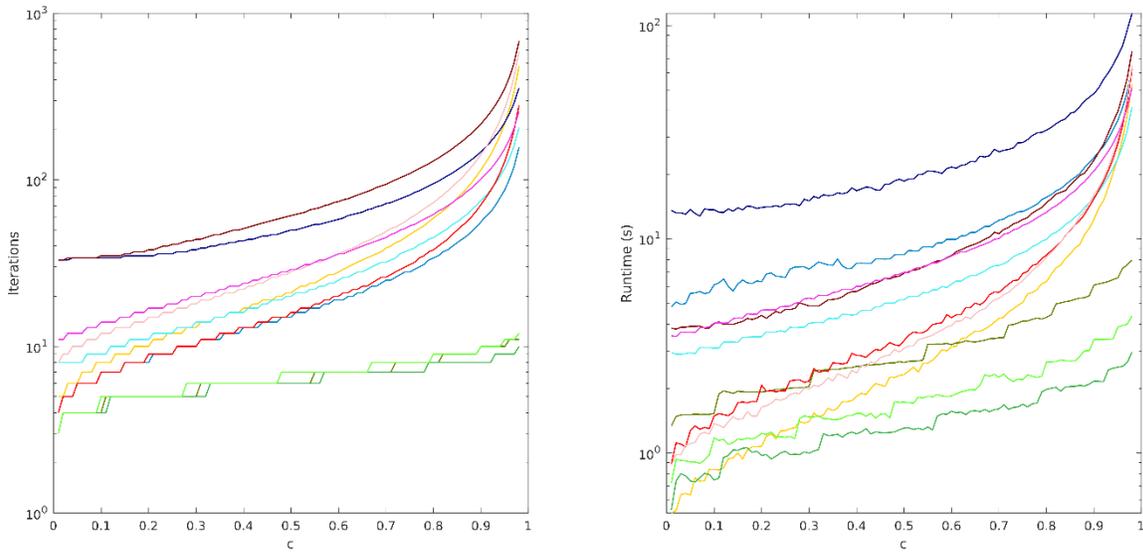


Figure A.58: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 1$

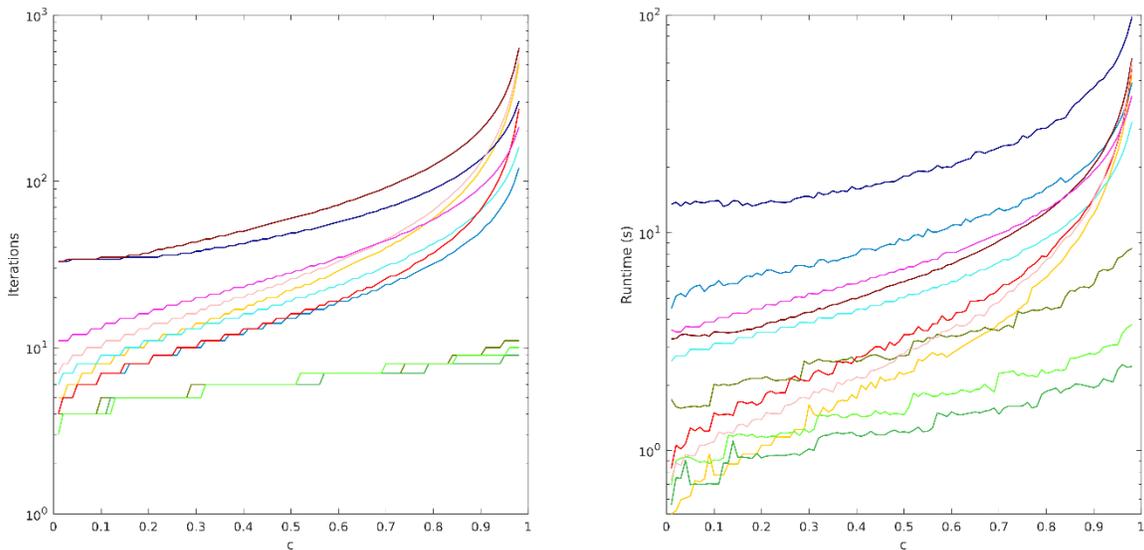


Figure A.59: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 2.5$

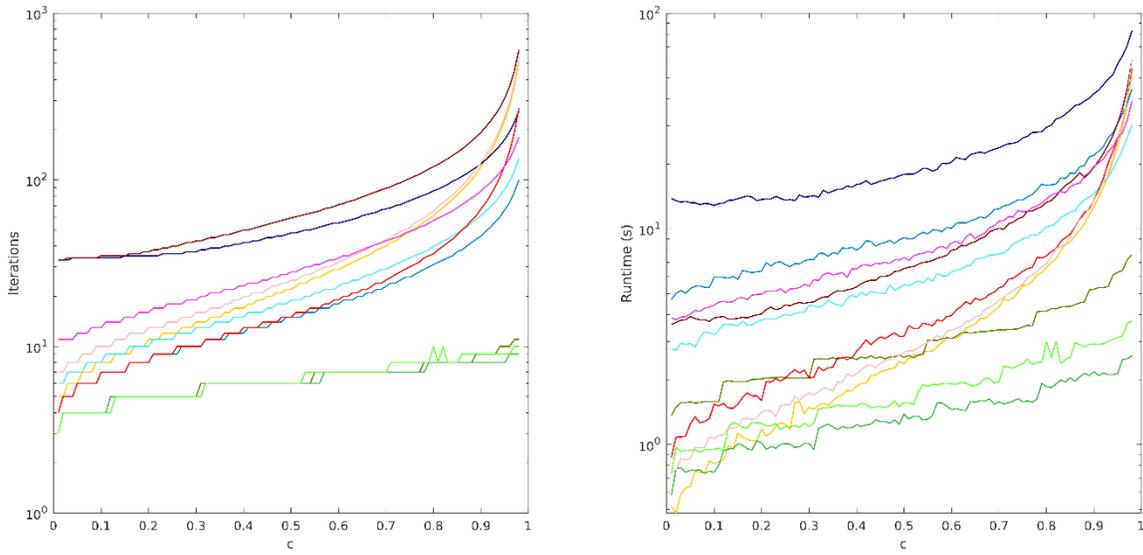


Figure A.60: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 5$

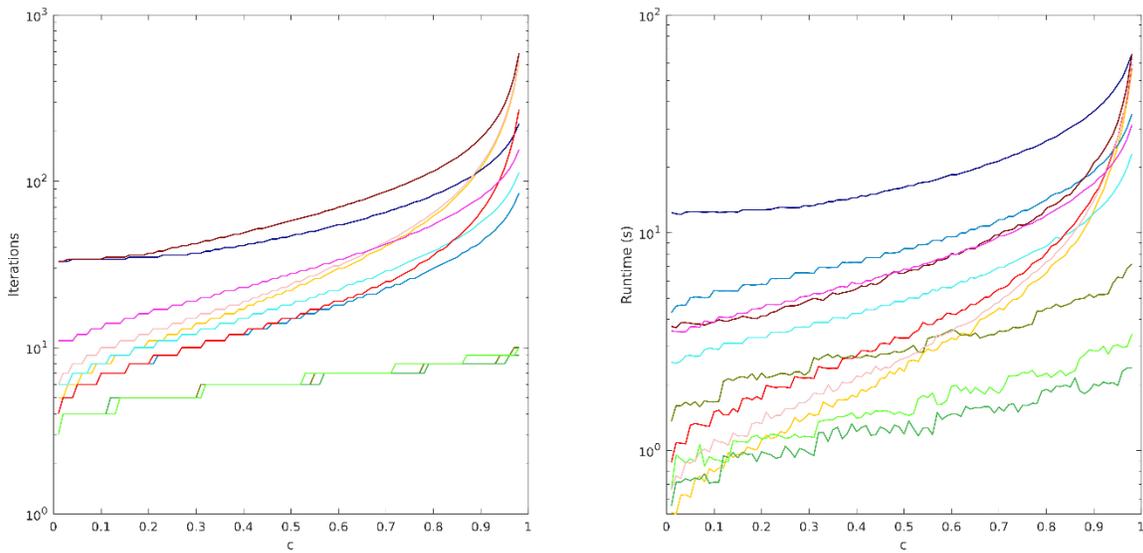


Figure A.61: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 10$

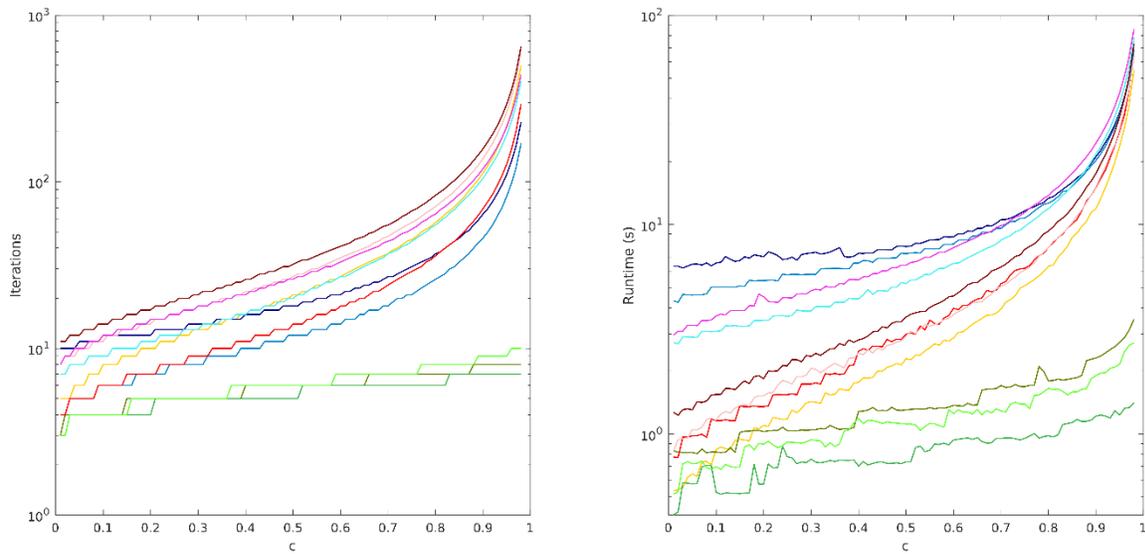


Figure A.62: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

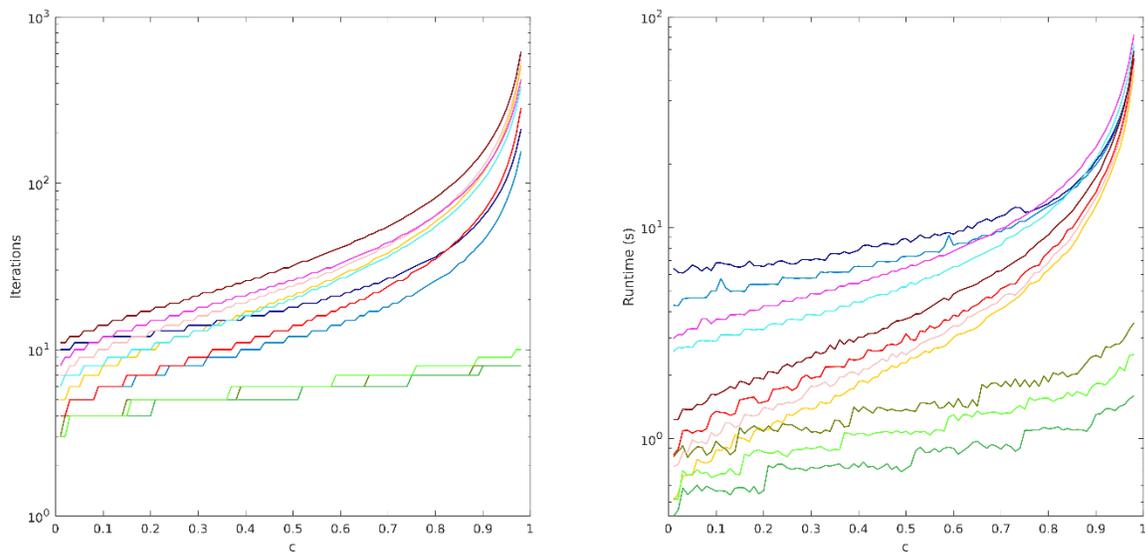


Figure A.63: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

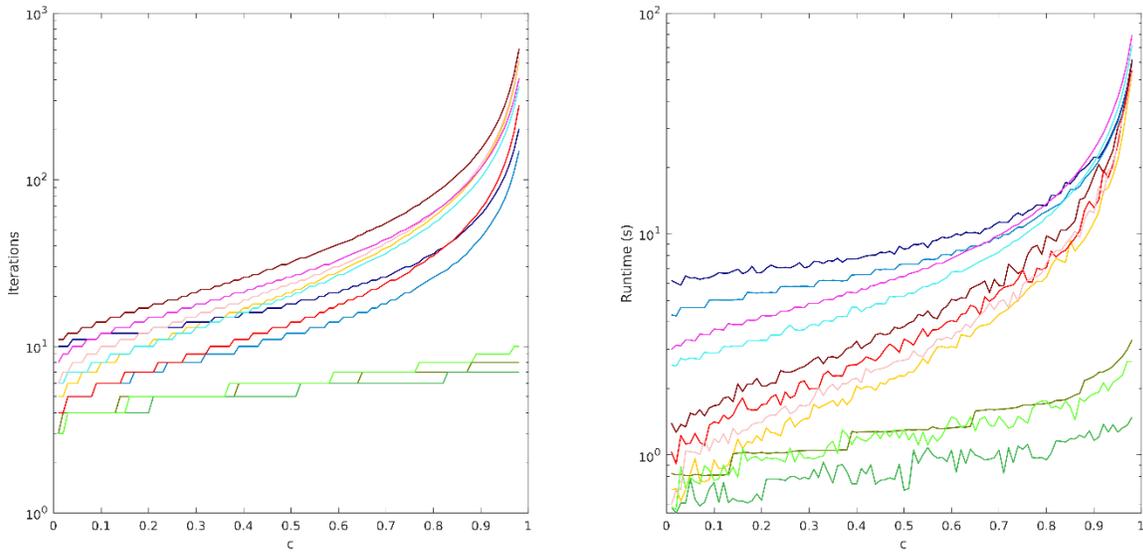


Figure A.64: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 5$

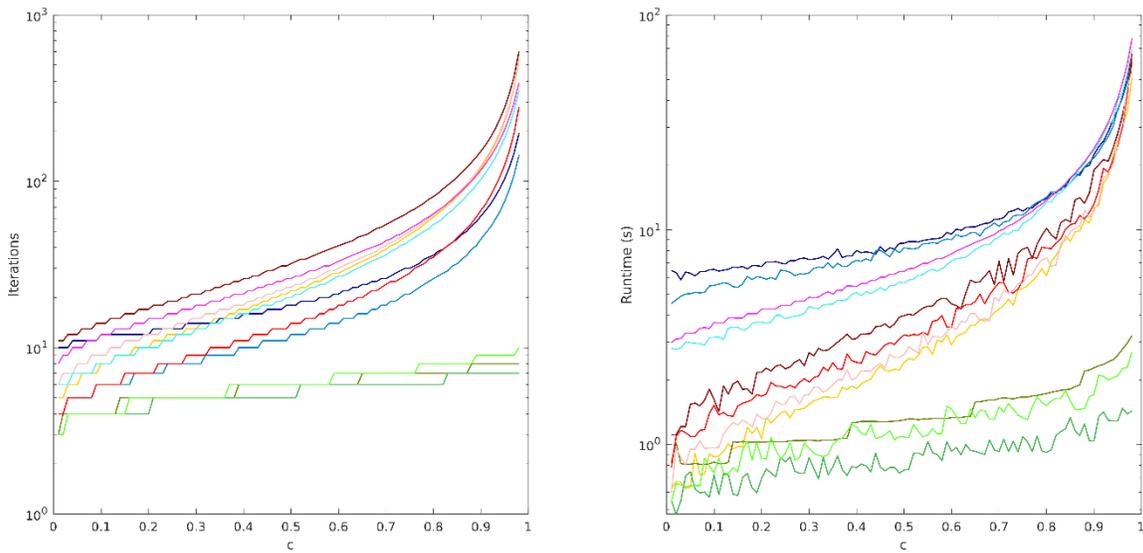


Figure A.65: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 10$

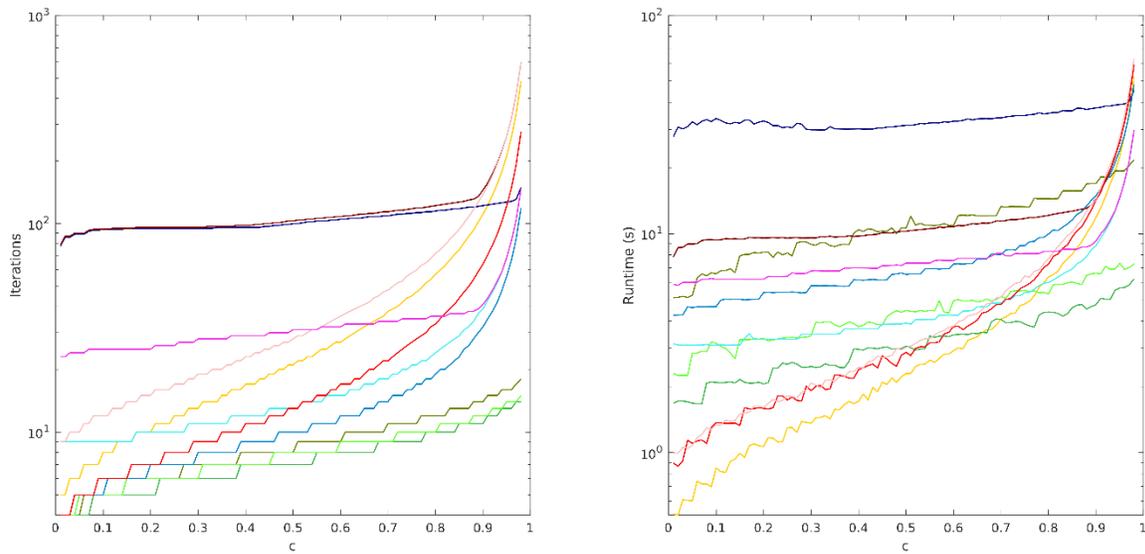


Figure A.66: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 1$

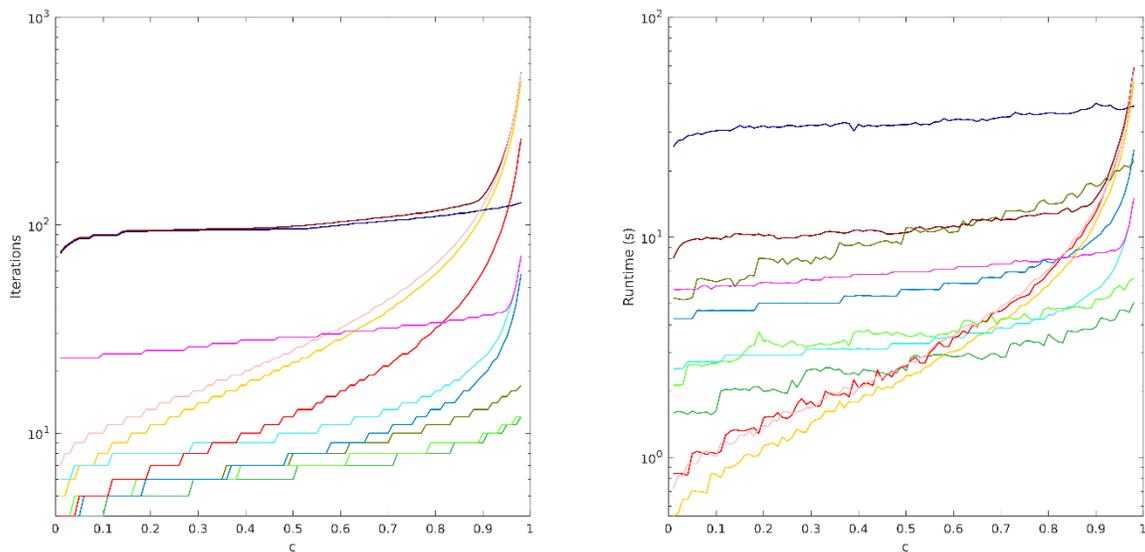


Figure A.67: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 2.5$

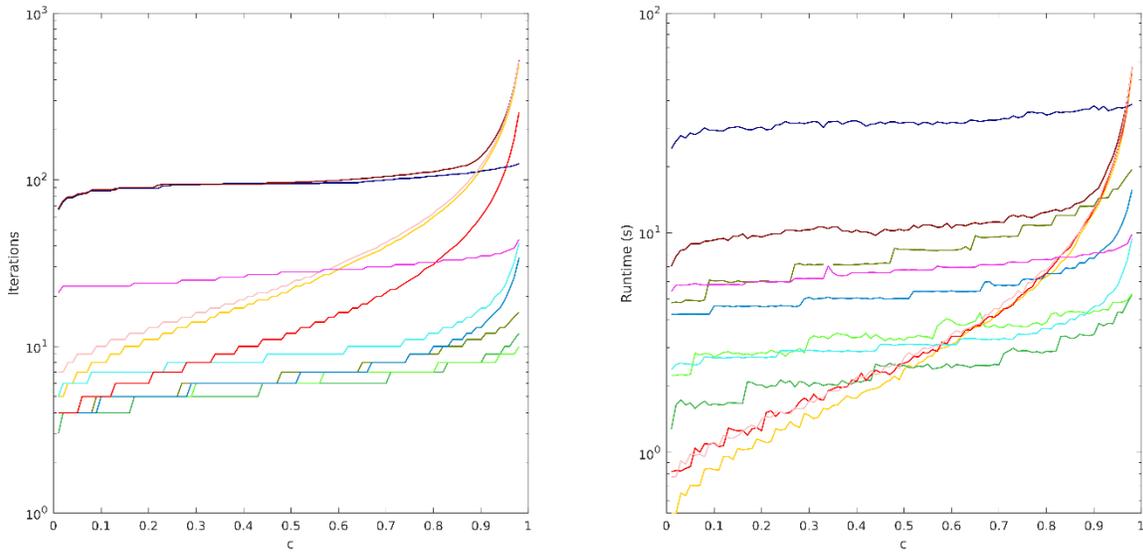


Figure A.68: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 5$

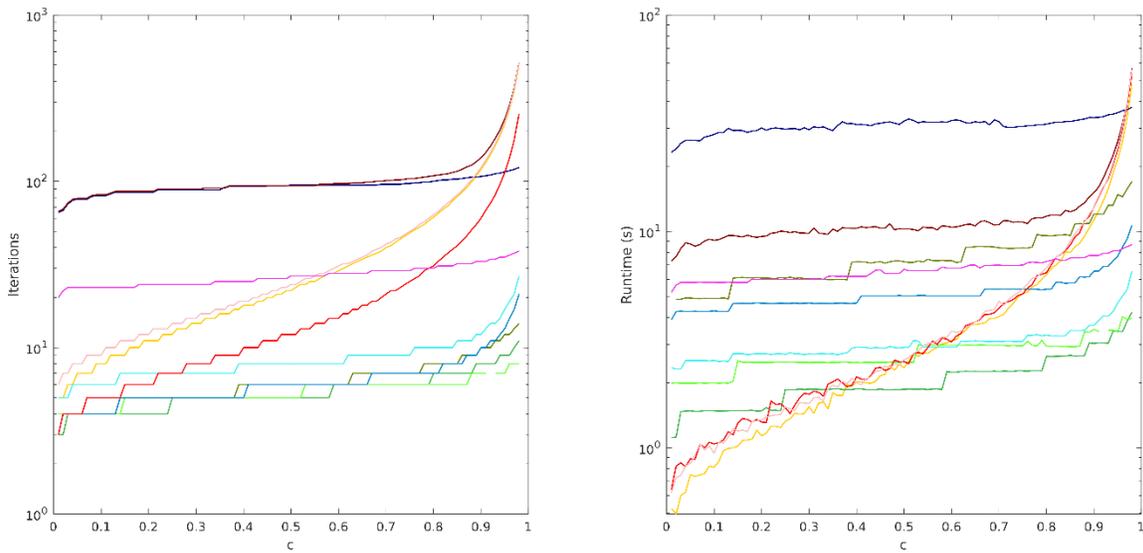


Figure A.69: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 10$

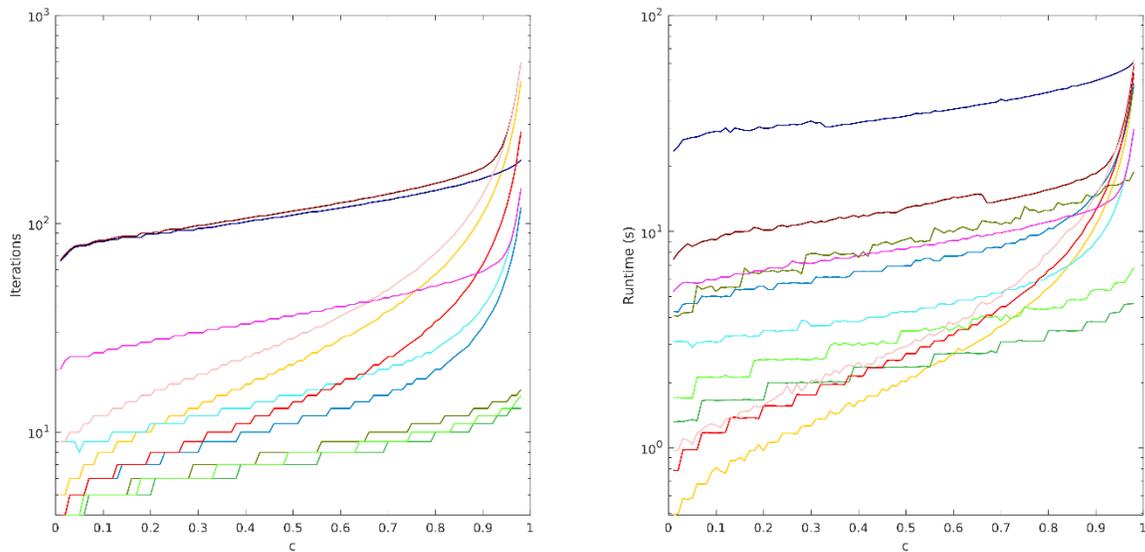


Figure A.70: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 1$

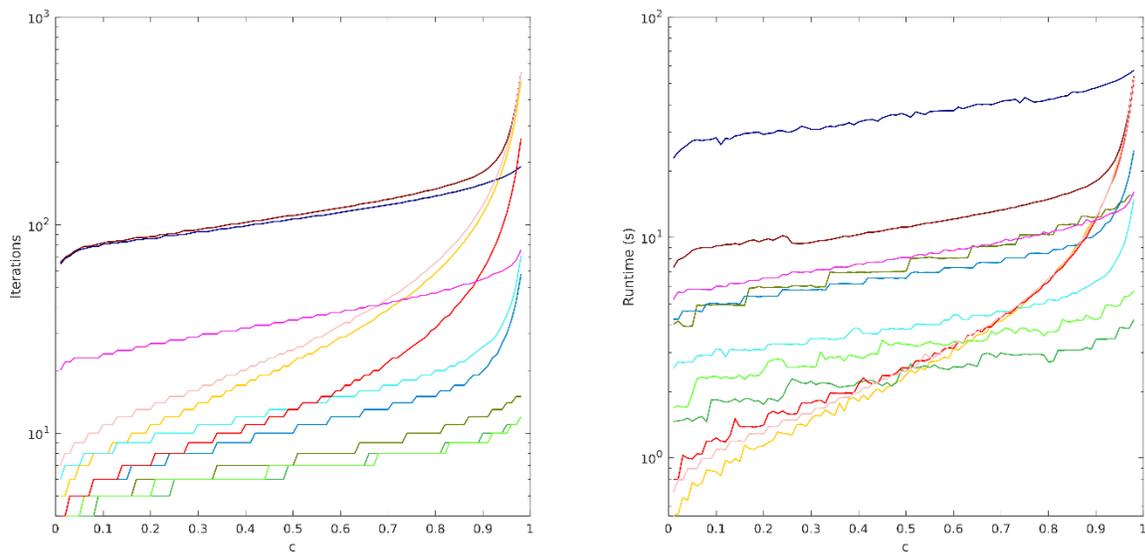


Figure A.71: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 2.5$

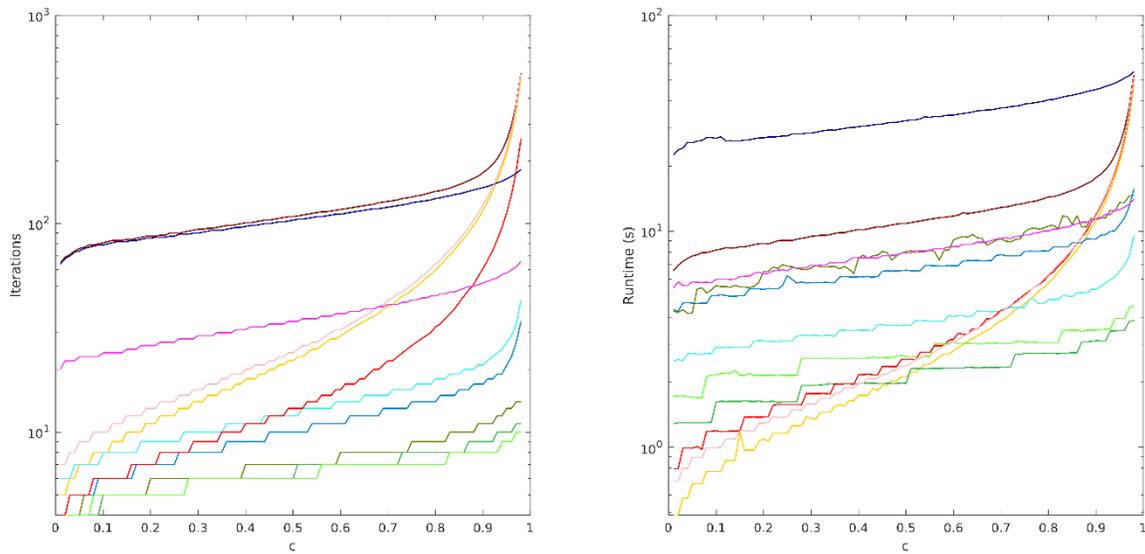


Figure A.72: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 5$

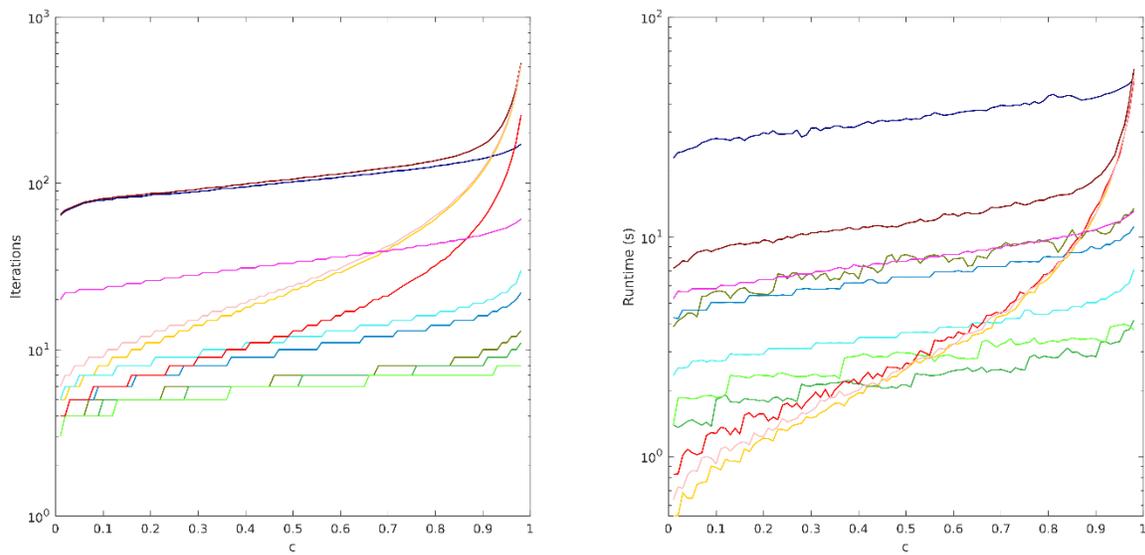


Figure A.73: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 10$

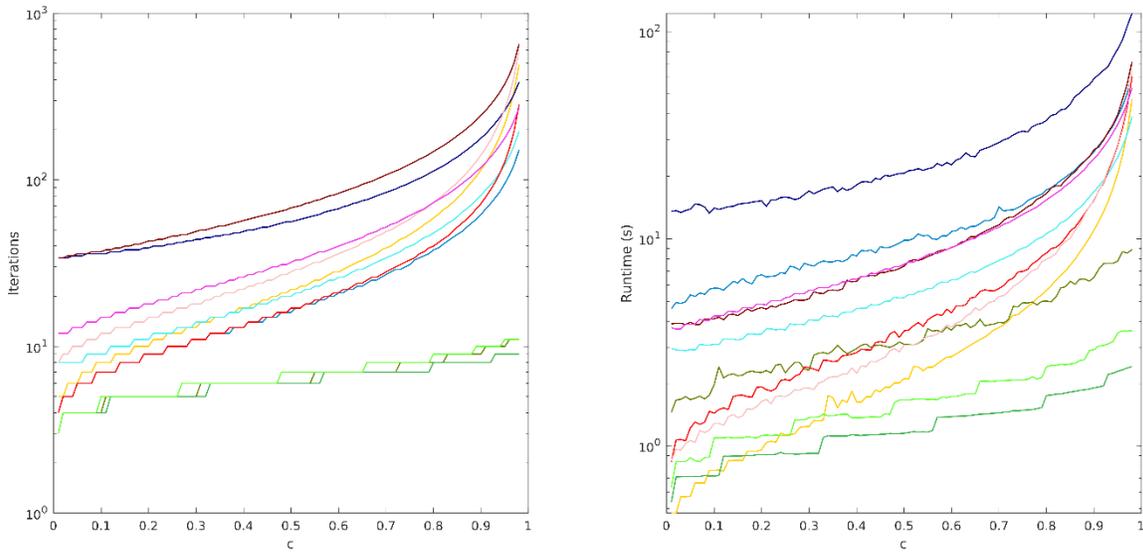


Figure A.74: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 1$

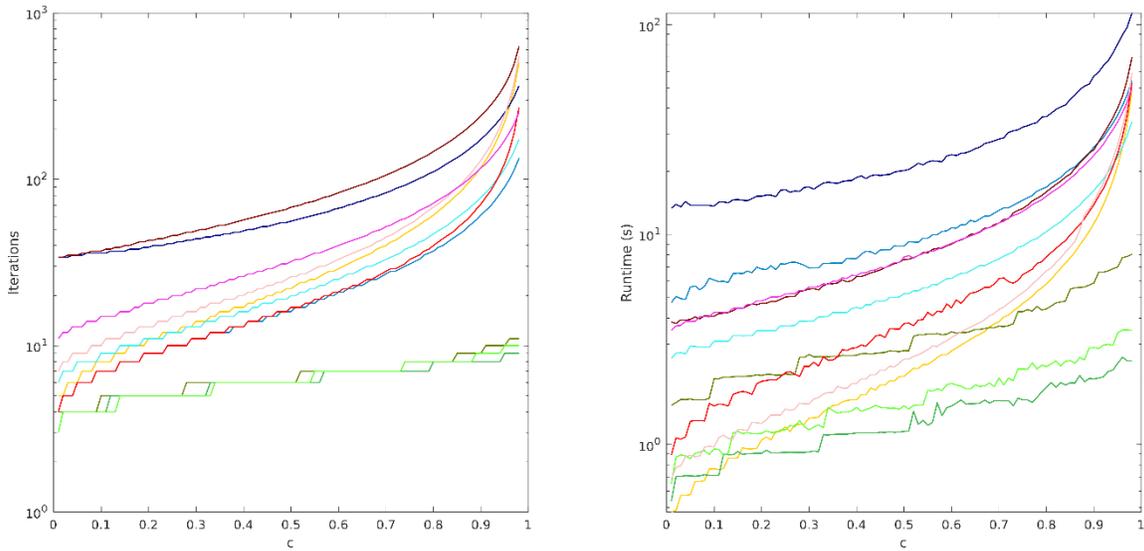


Figure A.75: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 2.5$

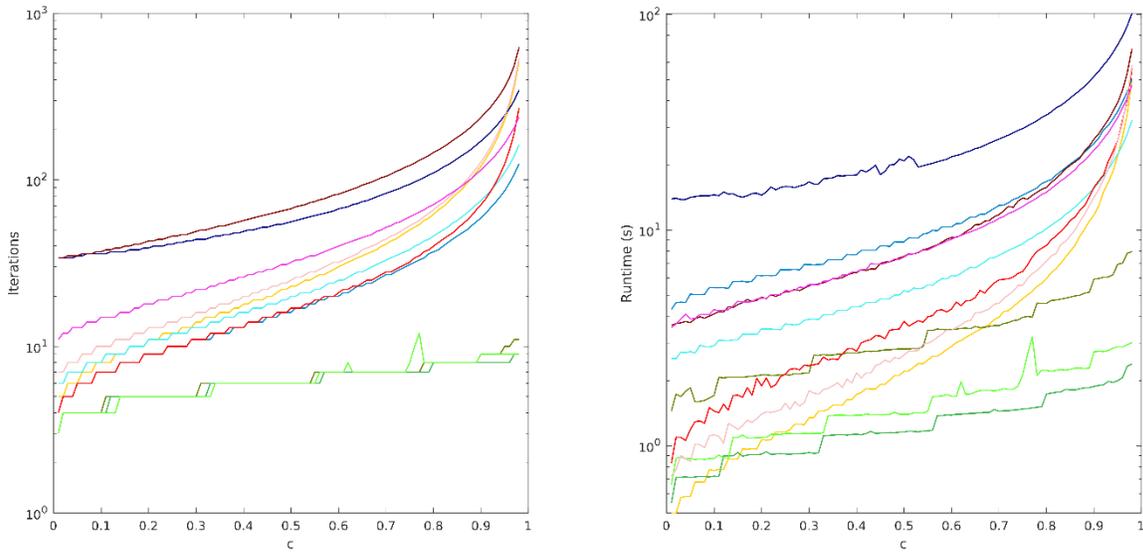


Figure A.76: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 5$

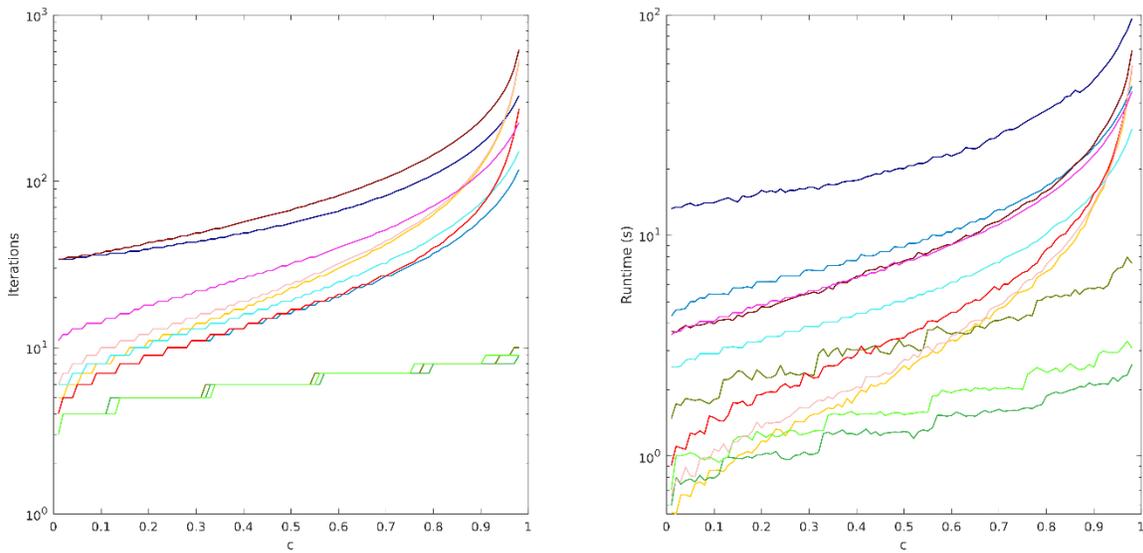


Figure A.77: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 10$

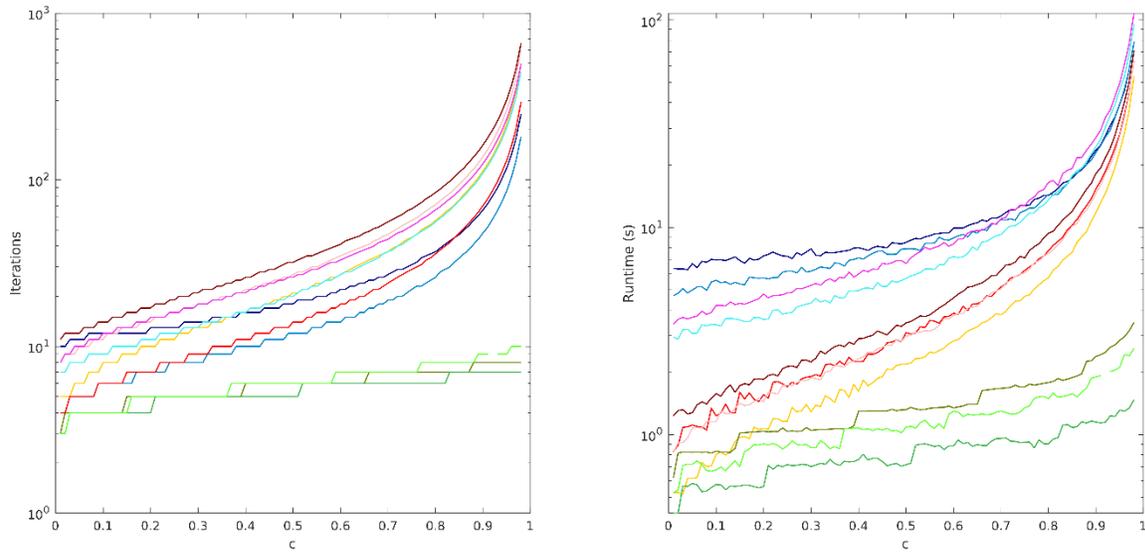


Figure A.78: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 1$

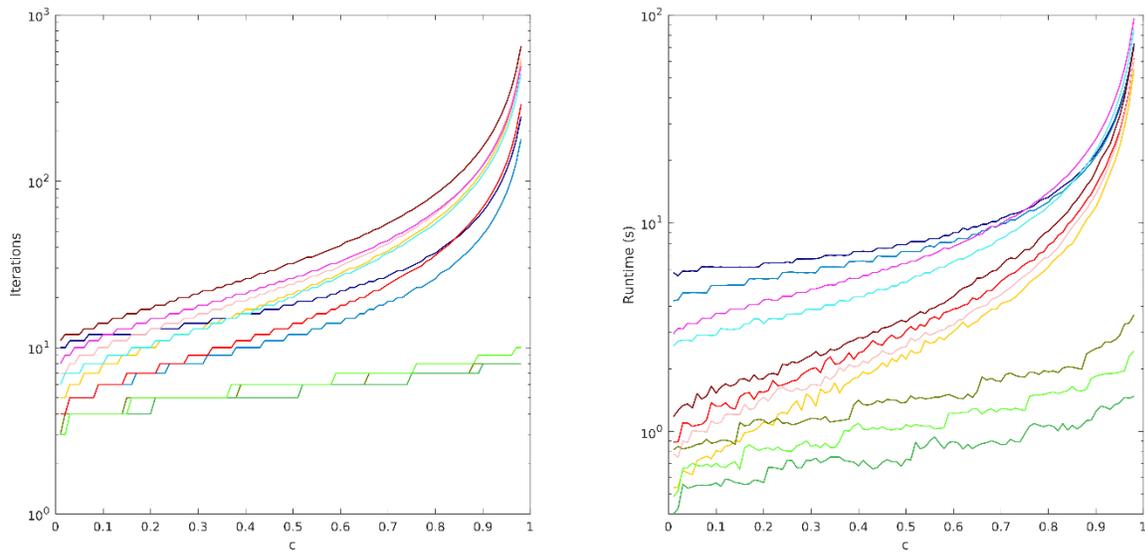


Figure A.79: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 2.5$

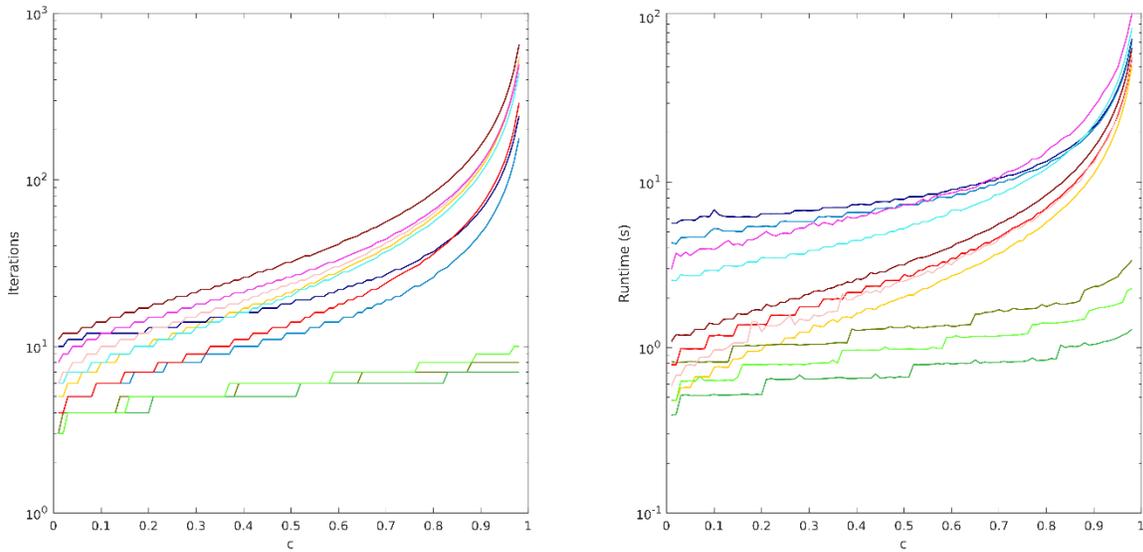


Figure A.80: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 5$

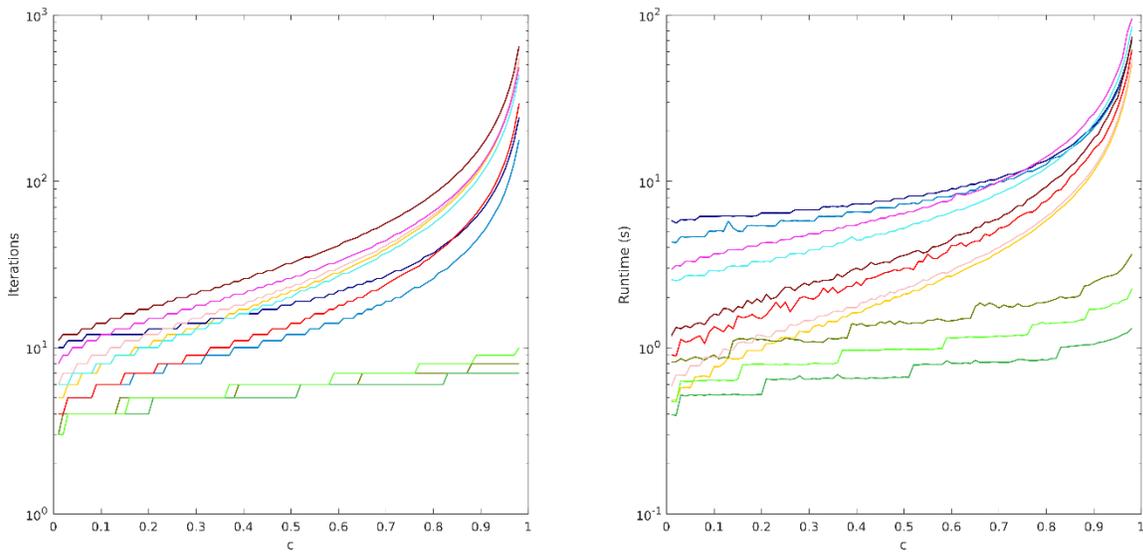


Figure A.81: Iterations required (left) and runtime observed (right) to converge parametric study problem versus scattering ratio:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 10$

## Appendix B: Full Results from Extreme Scattering Ratio Parametric Study in Section 4.7

Table B.1: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	771 / 74.06	N/C	N/C	N/C	N/C
<i>DSA</i>	101 / 212.02	194 / 497.02	228 / 569.99	232 / 591.99	N/A
<i>AP</i>	92 / 59.95	177 / 134.23	207 / 158.07	212 / 161.19	212 / 161.37
<i>pNDA</i>	95 / 137.54	153 / 241.20	164 / 259.85	165 / 261.57	165 / 262.99
<i>PBJ-ITMM</i>	430 / 125.92	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	206 / 81.26	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	332 / 67.59	988 / 197.84	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	461 / 89.66	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	339 / 68.69	N/C	N/C	N/C	N/C

Table B.2: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	889 / 96.62	N/C	N/C	N/C	N/C
<i>DSA</i>	66 / 136.84	186 / 464.45	267 / 711.12	284 / 769.06	N/A
<i>AP</i>	56 / 35.71	158 / 116.72	229 / 176.19	243 / 191.84	245 / 190.68
<i>pNDA</i>	55 / 71.74	135 / 248.93	163 / 312.22	167 / 276.84	167 / 309.63
<i>PBJ-ITMM</i>	229 / 74.00	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	115 / 51.03	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	175 / 38.79	815 / 179.00	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	495 / 107.84	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	179 / 37.08	830 / 166.16	N/C	N/C	N/C

Table B.3: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	916 / 90.51	N/C	N/C	N/C	N/C
<i>DSA</i>	42 / 79.93	153 / 339.63	264 / 658.58	297 / 788.99	N/A
<i>AP</i>	36 / 20.66	132 / 88.93	230 / 170.19	259 / 191.23	263 / 194.89
<i>pNDA</i>	31 / 28.81	103 / 165.58	144 / 247.07	150 / 265.12	152 / 255.85
<i>PBJ-ITMM</i>	137 / 41.94	730 / 211.39	N/C	N/C	N/C
<i>P-PI-SI</i>	68 / 28.68	385 / 149.43	N/C	N/C	N/C
<i>AH-PI-SI</i>	103 / 21.94	566 / 114.15	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	494 / 96.22	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	980 / 98.59	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	106 / 22.49	577 / 115.91	N/C	N/C	N/C

Table B.4: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	926 / 98.84	N/C	N/C	N/C	N/C
<i>DSA</i>	28 / 45.83	115 / 243.68	242 / 609.82	298 / 719.23	N/A
<i>AP</i>	25 / 13.00	102 / 67.08	217 / 157.92	267 / 202.31	275 / 205.05
<i>pNDA</i>	19 / 16.29	73 / 116.76	125 / 235.57	137 / 255.46	138 / 263.37
<i>PBJ-ITMM</i>	84 / 29.94	460 / 148.58	N/C	N/C	N/C
<i>P-PI-SI</i>	41 / 20.40	243 / 103.12	N/C	N/C	N/C
<i>AH-PI-SI</i>	62 / 13.74	355 / 72.10	N/C	N/C	N/C
<i>IPBJ</i>	969 / 108.61	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	485 / 103.45	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	958 / 96.98	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	65 / 14.33	363 / 73.48	N/C	N/C	N/C

Table B.5: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	777 / 84.25	N/C	N/C	N/C	N/C
<i>DSA</i>	39 / 49.92	65 / 100.56	75 / 117.22	77 / 123.07	N/A
<i>AP</i>	36 / 15.70	63 / 33.89	73 / 39.99	75 / 41.13	75 / 41.33
<i>pNDA</i>	40 / 29.50	52 / 44.67	58 / 51.02	58 / 50.40	59 / 51.50
<i>PBJ-ITMM</i>	430 / 134.90	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	212 / 90.82	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	339 / 73.09	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	466 / 98.12	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	346 / 70.03	N/C	N/C	N/C	N/C

Table B.6: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin} \Delta u_{thin} = 0.01, \Sigma_{thick} \Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	891 / 94.03	N/C	N/C	N/C	N/C
<i>DSA</i>	34 / 38.35	62 / 91.88	88 / 132.98	94 / 140.93	N/A
<i>AP</i>	30 / 11.68	60 / 28.10	84 / 41.95	89 / 44.27	90 / 45.25
<i>pNDA</i>	30 / 16.93	51 / 44.35	66 / 62.00	68 / 64.59	68 / 65.10
<i>PBJ-ITMM</i>	229 / 74.55	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	118 / 47.72	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	179 / 37.07	839 / 168.45	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	496 / 96.57	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	183 / 37.94	855 / 171.35	N/C	N/C	N/C

Table B.7: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	917 / 97.09	N/C	N/C	N/C	N/C
<i>DSA</i>	26 / 25.13	49 / 64.33	88 / 125.19	98 / 142.78	N/A
<i>AP</i>	23 / 7.74	47 / 20.74	84 / 41.48	94 / 45.79	95 / 45.00
<i>pNDA</i>	21 / 10.88	41 / 35.12	65 / 65.47	68 / 67.97	69 / 69.04
<i>PBJ-ITMM</i>	136 / 45.61	735 / 235.16	N/C	N/C	N/C
<i>P-PI-SI</i>	71 / 32.29	398 / 163.23	N/C	N/C	N/C
<i>AH-PI-SI</i>	105 / 22.56	581 / 117.24	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	494 / 105.14	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	980 / 99.30	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	109 / 23.43	593 / 118.92	N/C	N/C	N/C

Table B.8: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	926 / 100.31	N/C	N/C	N/C	N/C
<i>DSA</i>	20 / 15.40	48 / 56.16	81 / 118.37	99 / 147.81	N/A
<i>AP</i>	18 / 5.26	44 / 18.03	79 / 38.33	95 / 47.16	98 / 48.12
<i>pNDA</i>	15 / 6.54	34 / 25.47	61 / 60.87	67 / 67.81	68 / 70.52
<i>PBJ-ITMM</i>	84 / 29.67	462 / 144.35	N/C	N/C	N/C
<i>P-PI-SI</i>	42 / 21.25	251 / 108.65	N/C	N/C	N/C
<i>AH-PI-SI</i>	64 / 15.43	364 / 78.57	N/C	N/C	N/C
<i>IPBJ</i>	969 / 108.59	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	484 / 96.00	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	958 / 106.69	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	66 / 14.53	372 / 75.28	N/C	N/C	N/C

Table B.9: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	818 / 84.34	N/C	N/C	N/C	N/C
<i>DSA</i>	13 / 8.97	14 / 13.66	14 / 14.22	14 / 15.39	N/A
<i>AP</i>	12 / 3.49	13 / 4.88	13 / 4.77	13 / 4.61	13 / 4.61
<i>pNDA</i>	14 / 5.28	15 / 7.17	15 / 7.02	15 / 7.53	15 / 6.99
<i>PBJ-ITMM</i>	419 / 131.64	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	257 / 109.74	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	399 / 88.00	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	500 / 105.02	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	406 / 82.00	N/C	N/C	N/C	N/C

Table B.10: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	904 / 95.44	N/C	N/C	N/C	N/C
<i>DSA</i>	14 / 8.09	16 / 13.84	16 / 15.81	16 / 16.69	N/A
<i>AP</i>	12 / 3.06	15 / 5.12	15 / 5.67	16 / 6.06	16 / 6.12
<i>pNDA</i>	13 / 4.06	16 / 7.61	16 / 7.69	17 / 8.55	17 / 8.30
<i>PBJ-ITMM</i>	225 / 74.94	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	143 / 61.61	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	213 / 47.13	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	218 / 44.81	N/C	N/C	N/C	N/C

Table B.11: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	921 / 98.51	N/C	N/C	N/C	N/C
<i>DSA</i>	13 / 6.54	16 / 13.18	17 / 16.47	17 / 18.09	N/A
<i>AP</i>	12 / 2.82	15 / 4.59	16 / 6.11	17 / 6.44	17 / 6.18
<i>pNDA</i>	12 / 3.60	15 / 6.56	16 / 7.96	17 / 8.77	17 / 8.54
<i>PBJ-ITMM</i>	135 / 44.02	766 / 236.26	N/C	N/C	N/C
<i>P-PI-SI</i>	86 / 39.46	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	127 / 26.73	729 / 146.61	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	494 / 96.16	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	983 / 99.54	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	132 / 27.59	741 / 148.47	N/C	N/C	N/C

Table B.12: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	927 / 98.65	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 4.67	16 / 10.38	17 / 14.71	18 / 17.04	N/A
<i>AP</i>	11 / 1.98	15 / 3.79	17 / 5.57	17 / 5.72	17 / 5.91
<i>pNDA</i>	11 / 2.59	14 / 4.86	16 / 7.26	17 / 8.09	17 / 8.12
<i>PBJ-ITMM</i>	83 / 26.47	473 / 138.24	N/C	N/C	N/C
<i>P-PI-SI</i>	53 / 22.99	312 / 121.79	N/C	N/C	N/C
<i>AH-PI-SI</i>	79 / 17.21	450 / 90.91	N/C	N/C	N/C
<i>IPBJ</i>	970 / 96.42	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	481 / 93.86	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	959 / 96.74	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	82 / 17.66	459 / 92.43	N/C	N/C	N/C

Table B.13: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	891 / 94.31	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 4.03	8 / 6.37	8 / 6.75	8 / 6.71	N/A
<i>AP</i>	7 / 1.71	7 / 1.96	7 / 2.44	7 / 2.42	7 / 2.21
<i>pNDA</i>	11 / 3.23	11 / 4.23	11 / 4.29	11 / 4.29	11 / 4.22
<i>PBJ-ITMM</i>	345 / 108.69	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	271 / 113.76	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	565 / 114.06	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	573 / 115.33	N/C	N/C	N/C	N/C

Table B.14: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	919 / 96.53	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 3.23	8 / 5.74	8 / 7.09	8 / 7.03	N/A
<i>AP</i>	8 / 1.56	8 / 2.03	8 / 2.59	8 / 2.59	8 / 2.39
<i>pNDA</i>	11 / 2.74	11 / 3.90	11 / 4.43	11 / 4.47	11 / 4.52
<i>PBJ-ITMM</i>	199 / 65.71	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	162 / 68.40	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	333 / 75.32	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	339 / 74.65	N/C	N/C	N/C	N/C

Table B.15: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	923 / 88.46	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 2.48	9 / 4.74	9 / 6.38	9 / 7.14	N/A
<i>AP</i>	7 / 1.13	7 / 1.57	7 / 1.99	8 / 2.32	8 / 2.33
<i>pNDA</i>	10 / 2.07	10 / 2.97	11 / 4.01	11 / 4.05	11 / 4.05
<i>PBJ-ITMM</i>	125 / 38.48	790 / 228.57	N/C	N/C	N/C
<i>P-PI-SI</i>	102 / 41.88	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	214 / 44.07	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	488 / 94.84	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	981 / 99.37	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	220 / 45.91	N/C	N/C	N/C	N/C

Table B.16: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 4, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	928 / 89.87	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 2.37	9 / 4.19	9 / 6.83	9 / 7.99	N/A
<i>AP</i>	7 / 1.17	8 / 1.82	8 / 2.43	8 / 2.45	8 / 2.43
<i>pNDA</i>	9 / 1.87	10 / 2.76	10 / 3.62	10 / 3.81	10 / 3.95
<i>PBJ-ITMM</i>	80 / 28.24	481 / 153.36	N/C	N/C	N/C
<i>P-PI-SI</i>	65 / 29.94	398 / 162.19	N/C	N/C	N/C
<i>AH-PI-SI</i>	142 / 31.78	796 / 173.47	N/C	N/C	N/C
<i>IPBJ</i>	969 / 102.14	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	475 / 93.00	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	958 / 107.27	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	147 / 33.34	806 / 174.36	N/C	N/C	N/C

Table B.17: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	732 / 70.34	N/C	N/C	N/C	N/C
<i>DSA</i>	56 / 113.71	94 / 217.06	106 / 244.69	107 / 247.32	N/A
<i>AP</i>	50 / 29.79	85 / 58.97	96 / 68.56	97 / 67.78	97 / 67.81
<i>pNDA</i>	50 / 53.31	71 / 80.98	75 / 86.06	76 / 86.46	N/C
<i>PBJ-ITMM</i>	407 / 119.09	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	193 / 76.27	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	265 / 53.72	702 / 139.57	859 / 170.83	879 / 174.59	881 / 174.95
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	433 / 84.05	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	935 / 93.61	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	277 / 55.58	731 / 144.54	893 / 175.99	914 / 179.95	916 / 180.54

Table B.18: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	872 / 91.36	N/C	N/C	N/C	N/C
<i>DSA</i>	43 / 81.51	90 / 187.39	118 / 290.06	123 / 302.65	N/A
<i>AP</i>	35 / 20.64	75 / 50.47	99 / 68.04	103 / 70.40	104 / 73.41
<i>pNDA</i>	33 / 32.91	60 / 73.89	68 / 84.89	69 / 88.93	69 / 89.31
<i>PBJ-ITMM</i>	234 / 76.86	890 / 259.20	N/C	N/C	N/C
<i>P-PI-SI</i>	110 / 44.80	469 / 181.45	N/C	N/C	N/C
<i>AH-PI-SI</i>	145 / 30.57	594 / 118.40	958 / 190.14	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	472 / 91.72	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	970 / 97.28	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	152 / 31.19	619 / 122.68	996 / 196.04	N/C	N/C

Table B.19: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	918 / 94.58	N/C	N/C	N/C	N/C
<i>DSA</i>	34 / 56.99	77 / 159.77	116 / 279.11	126 / 300.30	N/A
<i>AP</i>	28 / 15.66	65 / 39.91	100 / 66.77	108 / 72.90	109 / 73.97
<i>pNDA</i>	23 / 18.85	47 / 58.30	59 / 79.47	60 / 78.61	60 / 79.45
<i>PBJ-ITMM</i>	153 / 52.06	653 / 210.96	N/C	N/C	N/C
<i>P-PI-SI</i>	66 / 29.51	345 / 145.81	N/C	N/C	N/C
<i>AH-PI-SI</i>	88 / 18.76	432 / 86.70	921 / 183.15	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	473 / 92.10	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	971 / 97.16	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	93 / 19.61	450 / 89.41	957 / 188.38	N/C	N/C

Table B.20: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.001, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	944 / 98.42	N/C	N/C	N/C	N/C
<i>DSA</i>	26 / 41.24	63 / 115.08	109 / 230.02	126 / 269.77	N/A
<i>AP</i>	22 / 9.24	55 / 30.48	96 / 59.91	112 / 69.37	114 / 70.43
<i>pNDA</i>	16 / 10.69	36 / 37.85	51 / 59.15	54 / 64.77	54 / 65.00
<i>PBJ-ITMM</i>	106 / 33.29	435 / 127.10	N/C	N/C	N/C
<i>P-PI-SI</i>	40 / 18.01	226 / 88.87	N/C	N/C	N/C
<i>AH-PI-SI</i>	55 / 12.22	282 / 56.88	818 / 162.50	N/C	N/C
<i>IPBJ</i>	989 / 98.49	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	480 / 93.57	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	972 / 97.47	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	59 / 13.07	294 / 58.87	851 / 167.86	N/C	N/C

Table B.21: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	742 / 81.25	N/C	N/C	N/C	N/C
<i>DSA</i>	26 / 37.11	43 / 71.55	48 / 81.15	48 / 80.22	N/A
<i>AP</i>	25 / 11.65	41 / 22.51	46 / 26.09	47 / 26.81	47 / 26.38
<i>pNDA</i>	24 / 18.47	36 / 31.56	38 / 34.10	39 / 34.21	39 / 34.67
<i>PBJ-ITMM</i>	410 / 131.06	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	201 / 86.55	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	274 / 61.02	743 / 163.81	918 / 199.48	941 / 207.06	943 / 199.35
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	440 / 96.09	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	946 / 95.21	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	286 / 63.24	773 / 164.18	954 / 202.10	978 / 209.16	980 / 207.06

Table B.22: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	876 / 83.99	N/C	N/C	N/C	N/C
<i>DSA</i>	23 / 26.78	41 / 59.26	53 / 78.18	55 / 83.82	N/A
<i>AP</i>	19 / 7.33	38 / 17.86	49 / 23.55	51 / 24.50	52 / 24.99
<i>pNDA</i>	18 / 10.29	34 / 26.75	39 / 32.01	39 / 31.97	40 / 32.89
<i>PBJ-ITMM</i>	235 / 70.10	913 / 263.89	N/C	N/C	N/C
<i>P-PI-SI</i>	114 / 46.24	493 / 190.68	N/C	N/C	N/C
<i>AH-PI-SI</i>	150 / 30.96	623 / 124.63	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	475 / 92.34	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	975 / 97.45	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	157 / 32.09	648 / 128.02	N/C	N/C	N/C

Table B.23: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	920 / 95.20	N/C	N/C	N/C	N/C
<i>DSA</i>	20 / 21.35	36 / 50.42	53 / 85.32	57 / 91.83	N/A
<i>AP</i>	17 / 6.29	34 / 16.51	49 / 25.97	53 / 27.31	54 / 27.61
<i>pNDA</i>	15 / 7.78	29 / 22.89	36 / 31.49	38 / 33.66	38 / 33.22
<i>PBJ-ITMM</i>	154 / 49.56	665 / 203.52	N/C	N/C	N/C
<i>P-PI-SI</i>	68 / 31.02	361 / 152.50	N/C	N/C	N/C
<i>AH-PI-SI</i>	91 / 20.58	450 / 94.99	980 / 206.53	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	475 / 101.67	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	973 / 98.27	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	96 / 21.68	468 / 99.99	N/C	N/C	N/C

Table B.24: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	945 / 90.56	N/C	N/C	N/C	N/C
<i>DSA</i>	18 / 15.77	29 / 37.28	50 / 73.04	57 / 84.17	N/A
<i>AP</i>	15 / 4.71	28 / 11.63	47 / 22.11	54 / 25.64	55 / 25.94
<i>pNDA</i>	12 / 5.25	23 / 15.09	34 / 28.13	36 / 30.49	36 / 30.20
<i>PBJ-ITMM</i>	106 / 33.04	440 / 130.70	N/C	N/C	N/C
<i>P-PI-SI</i>	41 / 18.45	234 / 91.94	N/C	N/C	N/C
<i>AH-PI-SI</i>	57 / 12.68	292 / 58.90	866 / 172.36	N/C	N/C
<i>IPBJ</i>	990 / 98.38	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	480 / 93.26	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	973 / 97.42	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	62 / 13.52	305 / 61.02	901 / 177.30	N/C	N/C

Table B.25: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	807 / 85.84	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 8.75	12 / 11.66	12 / 12.66	12 / 12.82	N/A
<i>AP</i>	10 / 2.77	11 / 3.79	11 / 3.89	11 / 3.87	11 / 3.91
<i>pNDA</i>	13 / 4.74	14 / 6.59	14 / 6.61	14 / 6.68	14 / 6.67
<i>PBJ-ITMM</i>	417 / 129.58	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	252 / 98.75	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	347 / 69.91	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	489 / 96.96	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	359 / 71.56	N/C	N/C	N/C	N/C

Table B.26: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	904 / 98.45	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 7.93	13 / 11.99	14 / 15.04	14 / 15.42	N/A
<i>AP</i>	10 / 2.66	12 / 4.19	12 / 4.58	12 / 4.53	12 / 4.62
<i>pNDA</i>	12 / 4.09	13 / 5.90	14 / 7.06	14 / 6.91	14 / 6.98
<i>PBJ-ITMM</i>	238 / 74.94	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	143 / 57.27	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	192 / 39.24	896 / 177.63	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	494 / 96.08	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	201 / 40.68	927 / 182.52	N/C	N/C	N/C

Table B.27: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	932 / 99.19	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 6.76	13 / 10.68	14 / 15.45	14 / 15.96	N/A
<i>AP</i>	10 / 2.53	12 / 4.02	12 / 4.63	13 / 5.10	13 / 5.11
<i>pNDA</i>	11 / 3.56	13 / 6.12	13 / 7.11	13 / 6.93	13 / 7.15
<i>PBJ-ITMM</i>	156 / 52.69	741 / 238.78	N/C	N/C	N/C
<i>P-PI-SI</i>	89 / 36.70	483 / 186.75	N/C	N/C	N/C
<i>AH-PI-SI</i>	122 / 25.45	619 / 123.22	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	485 / 94.34	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	985 / 98.69	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	129 / 26.69	640 / 126.41	N/C	N/C	N/C

Table B.28: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	949 / 99.52	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 5.95	13 / 9.58	14 / 14.21	14 / 15.69	N/A
<i>AP</i>	10 / 2.11	12 / 3.39	13 / 4.74	13 / 4.78	13 / 4.83
<i>pNDA</i>	10 / 2.95	12 / 4.95	13 / 6.77	13 / 7.15	13 / 7.00
<i>PBJ-ITMM</i>	109 / 36.82	474 / 149.45	N/C	N/C	N/C
<i>P-PI-SI</i>	58 / 26.99	305 / 127.61	N/C	N/C	N/C
<i>AH-PI-SI</i>	83 / 19.70	392 / 86.22	N/C	N/C	N/C
<i>IPBJ</i>	994 / 109.78	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	485 / 102.37	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	977 / 109.63	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	88 / 20.30	406 / 87.31	N/C	N/C	N/C

Table B.29: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	892 / 85.61	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 3.63	8 / 5.54	8 / 5.99	8 / 6.16	N/A
<i>AP</i>	7 / 1.41	7 / 1.86	7 / 1.97	7 / 2.01	7 / 2.00
<i>pNDA</i>	11 / 2.94	12 / 4.40	12 / 4.57	12 / 4.69	12 / 4.69
<i>PBJ-ITMM</i>	347 / 102.02	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	271 / 106.93	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	541 / 108.00	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	552 / 109.34	N/C	N/C	N/C	N/C

Table B.30: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	927 / 98.67	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 3.22	8 / 5.54	8 / 7.17	8 / 6.92	N/A
<i>AP</i>	8 / 1.47	8 / 2.11	8 / 2.38	8 / 2.51	8 / 2.42
<i>pNDA</i>	10 / 2.55	12 / 4.40	12 / 4.84	12 / 4.85	12 / 4.90
<i>PBJ-ITMM</i>	211 / 68.03	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	169 / 72.91	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	350 / 70.33	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	359 / 71.64	N/C	N/C	N/C	N/C

Table B.31: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	939 / 100.78	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 3.14	9 / 5.48	9 / 7.04	9 / 7.74	N/A
<i>AP</i>	7 / 1.32	7 / 1.77	7 / 2.23	7 / 2.37	7 / 2.19
<i>pNDA</i>	9 / 2.05	10 / 3.24	11 / 4.25	11 / 4.50	11 / 4.37
<i>PBJ-ITMM</i>	146 / 48.51	798 / 252.65	N/C	N/C	N/C
<i>P-PI-SI</i>	116 / 52.14	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	256 / 55.92	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	492 / 106.13	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	988 / 99.00	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	264 / 53.03	N/C	N/C	N/C	N/C

Table B.32: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 8, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	951 / 99.87	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 2.82	9 / 4.73	9 / 6.93	9 / 7.48	N/A
<i>AP</i>	7 / 1.16	7 / 1.68	7 / 2.11	7 / 2.19	7 / 2.21
<i>pNDA</i>	9 / 2.12	10 / 2.93	10 / 3.86	10 / 4.06	10 / 4.11
<i>PBJ-ITMM</i>	108 / 36.42	499 / 157.87	N/C	N/C	N/C
<i>P-PI-SI</i>	84 / 37.87	408 / 169.21	N/C	N/C	N/C
<i>AH-PI-SI</i>	198 / 40.46	798 / 158.57	N/C	N/C	N/C
<i>IPBJ</i>	995 / 105.69	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	488 / 95.12	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	978 / 97.94	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	205 / 41.48	815 / 160.59	N/C	N/C	N/C

Table B.33: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	708 / 75.61	N/C	N/C	N/C	N/C
<i>DSA</i>	34 / 64.72	47 / 102.07	51 / 110.97	51 / 111.49	N/A
<i>AP</i>	30 / 16.98	42 / 26.28	46 / 28.37	46 / 29.13	46 / 28.92
<i>pNDA</i>	30 / 23.30	38 / 31.96	40 / 33.53	40 / 33.86	40 / 33.21
<i>PBJ-ITMM</i>	378 / 123.21	843 / 267.83	994 / 318.03	N/C	N/C
<i>P-PI-SI</i>	180 / 79.27	458 / 195.91	N/C	N/C	N/C
<i>AH-PI-SI</i>	228 / 51.62	561 / 123.79	669 / 146.05	682 / 146.34	683 / 147.04
<i>IPBJ</i>	978 / 109.26	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	405 / 89.26	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	881 / 88.42	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	245 / 53.32	596 / 125.32	709 / 151.78	723 / 155.86	725 / 153.95

Table B.34: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	870 / 83.38	N/C	N/C	N/C	N/C
<i>DSA</i>	31 / 45.55	46 / 79.24	54 / 96.06	56 / 99.92	N/A
<i>AP</i>	24 / 11.38	38 / 21.17	45 / 25.32	46 / 25.89	46 / 25.91
<i>pNDA</i>	22 / 15.98	31 / 23.61	34 / 25.87	35 / 26.51	35 / 26.48
<i>PBJ-ITMM</i>	257 / 76.24	730 / 211.82	N/C	N/C	N/C
<i>P-PI-SI</i>	102 / 41.65	422 / 164.24	N/C	N/C	N/C
<i>AH-PI-SI</i>	128 / 26.44	486 / 96.49	742 / 146.55	785 / 155.12	790 / 155.98
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	461 / 89.51	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	956 / 95.03	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	140 / 28.52	515 / 101.14	783 / 152.87	829 / 161.79	834 / 162.73

Table B.35: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	930 / 100.57	N/C	N/C	N/C	N/C
<i>DSA</i>	27 / 41.11	43 / 77.08	54 / 101.46	56 / 105.33	N/A
<i>AP</i>	21 / 9.67	35 / 18.74	45 / 25.93	48 / 27.07	48 / 26.98
<i>pNDA</i>	17 / 10.38	25 / 18.62	29 / 21.79	29 / 22.30	29 / 21.85
<i>PBJ-ITMM</i>	195 / 58.42	563 / 164.13	N/C	N/C	N/C
<i>P-PI-SI</i>	61 / 26.29	316 / 123.12	N/C	N/C	N/C
<i>AH-PI-SI</i>	79 / 16.86	362 / 72.35	716 / 141.30	801 / 158.37	812 / 160.42
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	479 / 93.10	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	977 / 97.28	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	90 / 18.87	384 / 76.59	756 / 147.79	845 / 165.02	856 / 167.13

Table B.36: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	961 / 91.89	N/C	N/C	N/C	N/C
<i>DSA</i>	22 / 26.19	38 / 56.57	52 / 91.95	56 / 103.63	N/A
<i>AP</i>	18 / 6.41	32 / 15.74	45 / 25.11	49 / 27.47	50 / 28.14
<i>pNDA</i>	13 / 7.88	20 / 14.70	25 / 18.74	26 / 19.59	26 / 19.50
<i>PBJ-ITMM</i>	150 / 46.73	407 / 119.12	945 / 272.95	N/C	N/C
<i>P-PI-SI</i>	36 / 16.50	209 / 82.38	N/C	N/C	N/C
<i>AH-PI-SI</i>	50 / 11.22	242 / 48.69	645 / 127.40	794 / 169.56	815 / 160.86
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	488 / 94.77	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	985 / 98.11	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	62 / 13.41	259 / 51.47	681 / 133.18	838 / 163.47	860 / 167.85

Table B.37: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	722 / 77.96	N/C	N/C	N/C	N/C
<i>DSA</i>	21 / 29.14	28 / 45.12	30 / 47.46	30 / 49.30	N/A
<i>AP</i>	19 / 8.58	26 / 13.69	29 / 15.11	29 / 14.79	29 / 14.88
<i>pNDA</i>	20 / 12.56	25 / 17.47	26 / 18.61	26 / 19.39	26 / 18.93
<i>PBJ-ITMM</i>	390 / 121.99	900 / 283.49	N/C	N/C	N/C
<i>P-PI-SI</i>	188 / 84.10	493 / 203.68	N/C	N/C	N/C
<i>AH-PI-SI</i>	238 / 51.22	604 / 127.48	728 / 150.79	744 / 148.69	745 / 147.09
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	414 / 89.83	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	899 / 89.35	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	256 / 51.14	643 / 125.86	775 / 152.91	791 / 154.38	793 / 154.95

Table B.38: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	878 / 91.81	N/C	N/C	N/C	N/C
<i>DSA</i>	20 / 24.59	28 / 41.09	32 / 49.81	33 / 50.51	N/A
<i>AP</i>	17 / 6.91	25 / 12.31	29 / 14.55	30 / 15.09	30 / 14.78
<i>pNDA</i>	15 / 8.40	22 / 14.53	24 / 16.25	25 / 17.34	25 / 17.18
<i>PBJ-ITMM</i>	264 / 83.94	772 / 238.55	N/C	N/C	N/C
<i>P-PI-SI</i>	106 / 46.12	447 / 186.15	N/C	N/C	N/C
<i>AH-PI-SI</i>	133 / 29.33	516 / 108.36	805 / 168.21	856 / 179.15	862 / 180.15
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	466 / 98.61	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	965 / 102.52	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	147 / 33.01	548 / 118.57	853 / 184.26	906 / 197.05	912 / 196.36

Table B.39: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	935 / 89.62	N/C	N/C	N/C	N/C
<i>DSA</i>	18 / 19.48	26 / 32.47	32 / 45.16	34 / 47.70	N/A
<i>AP</i>	15 / 5.35	23 / 9.33	29 / 13.40	31 / 14.40	31 / 14.42
<i>pNDA</i>	13 / 6.39	19 / 11.33	22 / 14.22	23 / 14.88	23 / 14.95
<i>PBJ-ITMM</i>	200 / 59.93	590 / 171.69	N/C	N/C	N/C
<i>P-PI-SI</i>	64 / 27.63	331 / 128.88	N/C	N/C	N/C
<i>AH-PI-SI</i>	83 / 17.81	380 / 75.88	774 / 152.69	872 / 171.91	885 / 174.44
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	482 / 93.82	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	982 / 97.81	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	96 / 19.99	405 / 79.81	820 / 160.30	923 / 180.25	936 / 182.13

Table B.40: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16, \Sigma_{thin}\Delta u_{thin} = 0.01, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	963 / 101.54	N/C	N/C	N/C	N/C
<i>DSA</i>	16 / 16.97	23 / 30.05	31 / 48.21	34 / 54.02	N/A
<i>AP</i>	14 / 4.77	21 / 9.01	29 / 14.51	31 / 15.67	32 / 16.06
<i>pNDA</i>	11 / 5.40	16 / 10.12	20 / 13.35	21 / 14.95	21 / 14.75
<i>PBJ-ITMM</i>	153 / 49.89	423 / 132.91	N/C	N/C	N/C
<i>P-PI-SI</i>	39 / 18.91	217 / 89.58	N/C	N/C	N/C
<i>AH-PI-SI</i>	54 / 12.05	252 / 50.87	692 / 138.55	864 / 170.99	889 / 176.55
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	489 / 95.76	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	988 / 98.94	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	68 / 14.61	272 / 54.27	734 / 144.04	914 / 179.22	941 / 186.04

Table B.41: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	806 / 77.27	N/C	N/C	N/C	N/C
<i>DSA</i>	11 / 8.37	12 / 11.32	12 / 11.98	12 / 12.16	N/A
<i>AP</i>	10 / 2.73	10 / 3.25	11 / 3.76	11 / 3.81	11 / 3.73
<i>pNDA</i>	12 / 4.37	13 / 5.63	13 / 5.95	13 / 5.80	13 / 5.98
<i>PBJ-ITMM</i>	432 / 126.18	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	248 / 97.26	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	322 / 64.44	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	476 / 92.56	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	993 / 98.66	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	349 / 68.98	N/C	N/C	N/C	N/C

Table B.42: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	917 / 88.47	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 8.11	13 / 11.80	13 / 13.21	13 / 13.14	N/A
<i>AP</i>	10 / 2.53	11 / 3.55	11 / 3.93	11 / 3.93	11 / 3.94
<i>pNDA</i>	11 / 3.63	12 / 5.25	13 / 6.11	13 / 6.31	13 / 6.34
<i>PBJ-ITMM</i>	291 / 86.40	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	148 / 59.43	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	194 / 39.63	801 / 159.22	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	494 / 96.58	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	219 / 43.80	859 / 167.67	N/C	N/C	N/C

Table B.43: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	955 / 97.62	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 8.59	13 / 12.15	13 / 13.93	13 / 13.97	N/A
<i>AP</i>	10 / 2.43	11 / 3.60	11 / 4.03	11 / 4.07	11 / 4.20
<i>pNDA</i>	10 / 3.39	11 / 4.86	12 / 5.83	12 / 5.94	12 / 6.03
<i>PBJ-ITMM</i>	222 / 72.28	743 / 235.22	N/C	N/C	N/C
<i>P-PI-SI</i>	102 / 42.07	465 / 183.74	N/C	N/C	N/C
<i>AH-PI-SI</i>	140 / 29.08	563 / 111.54	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	497 / 96.55	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	162 / 32.86	608 / 119.17	N/C	N/C	N/C

Table B.44: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	973 / 103.01	N/C	N/C	N/C	N/C
<i>DSA</i>	11 / 6.54	13 / 10.58	13 / 13.30	13 / 13.76	N/A
<i>AP</i>	10 / 2.50	11 / 3.16	11 / 3.86	12 / 4.57	12 / 4.55
<i>pNDA</i>	9 / 2.99	11 / 4.40	11 / 5.39	11 / 5.40	11 / 5.56
<i>PBJ-ITMM</i>	172 / 55.70	511 / 157.54	N/C	N/C	N/C
<i>P-PI-SI</i>	76 / 31.70	299 / 116.65	N/C	N/C	N/C
<i>AH-PI-SI</i>	105 / 22.05	369 / 73.53	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	497 / 96.54	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	998 / 99.13	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	123 / 25.35	404 / 79.62	N/C	N/C	N/C

Table B.45: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	897 / 94.04	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 3.97	8 / 6.05	8 / 6.33	8 / 6.23	N/A
<i>AP</i>	7 / 1.39	7 / 1.86	7 / 2.01	7 / 2.01	7 / 2.01
<i>pNDA</i>	10 / 2.67	11 / 3.84	12 / 4.39	12 / 4.41	12 / 4.45
<i>PBJ-ITMM</i>	355 / 106.10	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	275 / 107.55	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	565 / 112.09	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	591 / 115.98	N/C	N/C	N/C	N/C

Table B.46: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	945 / 100.88	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 3.64	8 / 5.81	8 / 7.09	8 / 7.36	N/A
<i>AP</i>	8 / 1.57	8 / 2.09	8 / 2.40	8 / 2.61	8 / 2.46
<i>pNDA</i>	10 / 2.86	12 / 4.53	12 / 4.75	12 / 4.92	12 / 4.85
<i>PBJ-ITMM</i>	255 / 82.28	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	195 / 77.09	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	446 / 88.57	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	470 / 92.42	N/C	N/C	N/C	N/C

Table B.47: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	964 / 103.48	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 3.51	8 / 5.16	9 / 7.53	9 / 8.02	N/A
<i>AP</i>	7 / 1.37	7 / 1.81	7 / 2.22	7 / 2.40	7 / 2.29
<i>pNDA</i>	10 / 2.53	10 / 3.45	10 / 3.94	11 / 4.78	11 / 5.00
<i>PBJ-ITMM</i>	211 / 72.12	846 / 269.40	N/C	N/C	N/C
<i>P-PI-SI</i>	160 / 69.04	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	382 / 80.87	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	403 / 85.47	N/C	N/C	N/C	N/C

Table B.48: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 16$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	974 / 93.32	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 3.10	9 / 4.64	9 / 6.46	9 / 7.22	N/A
<i>AP</i>	7 / 1.26	7 / 1.56	7 / 1.94	7 / 2.05	7 / 2.10
<i>pNDA</i>	10 / 2.27	10 / 2.84	10 / 3.42	10 / 3.57	10 / 3.61
<i>PBJ-ITMM</i>	181 / 54.50	579 / 168.28	N/C	N/C	N/C
<i>P-PI-SI</i>	136 / 54.61	456 / 176.52	N/C	N/C	N/C
<i>AH-PI-SI</i>	331 / 66.14	971 / 191.36	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	500 / 97.24	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	998 / 99.17	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	351 / 69.43	N/C	N/C	N/C	N/C

Table B.49: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	761 / 80.10	N/C	N/C	N/C	N/C
<i>DSA</i>	26 / 38.78	32 / 48.40	33 / 50.46	33 / 50.38	N/A
<i>AP</i>	23 / 10.73	28 / 13.61	29 / 14.13	29 / 14.17	29 / 14.33
<i>pNDA</i>	23 / 13.76	27 / 17.30	27 / 17.49	28 / 18.55	28 / 18.10
<i>PBJ-ITMM</i>	353 / 110.38	768 / 240.43	948 / 288.56	972 / 300.48	975 / 297.49
<i>P-PI-SI</i>	185 / 73.24	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	230 / 45.48	655 / 126.81	823 / 159.04	845 / 163.28	848 / 163.76
<i>IPBJ</i>	970 / 96.68	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	435 / 84.58	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	936 / 91.58	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	239 / 47.54	670 / 130.63	842 / 163.94	864 / 168.06	867 / 168.56

Table B.50: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	893 / 93.31	N/C	N/C	N/C	N/C
<i>DSA</i>	25 / 35.93	32 / 49.26	35 / 54.56	35 / 53.89	N/A
<i>AP</i>	19 / 8.53	26 / 12.68	28 / 13.68	28 / 13.77	28 / 13.86
<i>pNDA</i>	17 / 10.62	22 / 13.88	23 / 14.98	23 / 14.78	23 / 15.23
<i>PBJ-ITMM</i>	287 / 86.62	643 / 200.17	N/C	N/C	N/C
<i>P-PI-SI</i>	100 / 43.78	474 / 195.32	N/C	N/C	N/C
<i>AH-PI-SI</i>	123 / 25.14	562 / 108.96	997 / 192.21	N/C	N/C
<i>IPBJ</i>	992 / 106.05	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	473 / 98.52	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	976 / 95.45	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	133 / 27.16	571 / 111.41	N/C	N/C	N/C

Table B.51: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	927 / 99.64	N/C	N/C	N/C	N/C
<i>DSA</i>	23 / 33.72	31 / 45.69	35 / 52.91	35 / 54.35	N/A
<i>AP</i>	18 / 8.18	25 / 11.67	29 / 14.25	29 / 14.57	29 / 14.71
<i>pNDA</i>	13 / 7.88	18 / 11.07	19 / 12.45	20 / 12.99	20 / 13.11
<i>PBJ-ITMM</i>	241 / 77.90	484 / 151.94	N/C	N/C	N/C
<i>P-PI-SI</i>	57 / 26.39	327 / 139.80	N/C	N/C	N/C
<i>AH-PI-SI</i>	73 / 15.42	392 / 76.46	948 / 185.60	N/C	N/C
<i>IPBJ</i>	984 / 106.17	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	478 / 93.11	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	971 / 95.00	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	93 / 19.36	399 / 78.58	960 / 186.52	N/C	N/C

Table B.52: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	943 / 100.30	N/C	N/C	N/C	N/C
<i>DSA</i>	20 / 29.15	29 / 42.45	34 / 52.76	35 / 53.61	N/A
<i>AP</i>	16 / 7.13	24 / 11.10	29 / 14.86	30 / 14.90	31 / 15.61
<i>pNDA</i>	10 / 6.01	15 / 9.35	17 / 10.86	17 / 10.76	17 / 11.23
<i>PBJ-ITMM</i>	198 / 65.45	382 / 124.43	894 / 289.46	N/C	N/C
<i>P-PI-SI</i>	33 / 17.26	201 / 86.75	N/C	N/C	N/C
<i>AH-PI-SI</i>	46 / 10.26	246 / 48.57	814 / 157.16	N/C	N/C
<i>IPBJ</i>	981 / 108.80	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	479 / 104.91	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	966 / 94.53	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	71 / 15.05	253 / 50.20	825 / 160.79	N/C	N/C

Table B.53: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	769 / 81.26	N/C	N/C	N/C	N/C
<i>DSA</i>	19 / 23.10	22 / 28.54	23 / 31.16	23 / 30.89	N/A
<i>AP</i>	17 / 7.24	20 / 8.80	21 / 9.30	21 / 9.42	21 / 9.32
<i>pNDA</i>	18 / 9.70	20 / 11.27	21 / 11.74	21 / 12.10	21 / 11.85
<i>PBJ-ITMM</i>	385 / 120.74	822 / 252.88	N/C	N/C	N/C
<i>P-PI-SI</i>	188 / 80.30	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	234 / 46.21	677 / 130.95	858 / 165.56	882 / 170.16	884 / 170.47
<i>IPBJ</i>	989 / 106.84	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	440 / 85.56	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	946 / 94.24	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	247 / 49.07	698 / 136.03	883 / 171.79	908 / 176.62	911 / 176.96

Table B.54: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	898 / 93.47	N/C	N/C	N/C	N/C
<i>DSA</i>	18 / 22.51	22 / 27.90	24 / 31.15	24 / 30.37	N/A
<i>AP</i>	15 / 5.97	19 / 7.87	21 / 9.07	21 / 9.30	21 / 9.11
<i>pNDA</i>	14 / 7.01	18 / 9.62	19 / 10.47	19 / 10.50	19 / 10.60
<i>PBJ-ITMM</i>	313 / 99.22	686 / 212.32	N/C	N/C	N/C
<i>P-PI-SI</i>	101 / 44.58	486 / 200.61	N/C	N/C	N/C
<i>AH-PI-SI</i>	125 / 25.35	574 / 111.30	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	476 / 93.81	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	981 / 96.03	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	145 / 29.44	587 / 114.61	N/C	N/C	N/C

Table B.55: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	931 / 97.60	N/C	N/C	N/C	N/C
<i>DSA</i>	17 / 21.19	22 / 27.97	24 / 32.16	25 / 33.22	N/A
<i>AP</i>	14 / 5.73	19 / 7.64	21 / 9.09	22 / 9.76	22 / 9.49
<i>pNDA</i>	11 / 5.89	16 / 8.77	17 / 9.67	17 / 9.43	17 / 9.02
<i>PBJ-ITMM</i>	259 / 76.84	531 / 154.57	N/C	N/C	N/C
<i>P-PI-SI</i>	59 / 25.24	335 / 130.42	N/C	N/C	N/C
<i>AH-PI-SI</i>	76 / 16.14	398 / 77.77	976 / 188.63	N/C	N/C
<i>IPBJ</i>	992 / 98.50	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	480 / 93.30	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	975 / 95.50	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	107 / 22.13	409 / 80.30	993 / 193.28	N/C	N/C

Table B.56: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ ,  $\Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	947 / 101.10	N/C	N/C	N/C	N/C
<i>DSA</i>	15 / 17.14	20 / 22.97	24 / 28.34	25 / 29.84	N/A
<i>AP</i>	13 / 4.82	18 / 6.63	21 / 8.25	22 / 8.83	22 / 8.84
<i>pNDA</i>	10 / 4.78	14 / 6.86	16 / 8.23	16 / 8.11	16 / 8.12
<i>PBJ-ITMM</i>	212 / 63.29	423 / 124.51	945 / 272.65	N/C	N/C
<i>P-PI-SI</i>	38 / 17.27	206 / 81.25	N/C	N/C	N/C
<i>AH-PI-SI</i>	52 / 11.37	250 / 50.06	834 / 161.05	N/C	N/C
<i>IPBJ</i>	989 / 98.07	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	481 / 93.52	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	970 / 94.99	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	84 / 17.64	261 / 51.85	850 / 166.77	N/C	N/C

Table B.57: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	829 / 88.53	N/C	N/C	N/C	N/C
<i>DSA</i>	11 / 9.13	12 / 11.18	12 / 11.36	12 / 11.16	N/A
<i>AP</i>	10 / 3.08	10 / 3.23	10 / 3.61	10 / 3.55	10 / 3.38
<i>pNDA</i>	15 / 6.10	13 / 5.73	13 / 6.05	13 / 5.93	13 / 5.91
<i>PBJ-ITMM</i>	501 / 160.99	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	246 / 96.57	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	315 / 61.79	978 / 188.57	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	482 / 93.75	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	376 / 74.13	N/C	N/C	N/C	N/C

Table B.58: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	928 / 99.59	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 9.81	13 / 11.57	13 / 12.05	13 / 12.41	N/A
<i>AP</i>	10 / 3.06	10 / 3.21	11 / 3.66	11 / 3.63	11 / 3.53
<i>pNDA</i>	11 / 4.55	12 / 5.51	12 / 5.41	12 / 5.71	12 / 5.72
<i>PBJ-ITMM</i>	411 / 133.93	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	172 / 73.59	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	227 / 44.91	756 / 145.97	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	497 / 96.84	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	292 / 57.71	836 / 162.49	N/C	N/C	N/C

Table B.59: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	951 / 103.11	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 8.60	13 / 10.56	13 / 11.18	13 / 11.50	N/A
<i>AP</i>	10 / 2.61	11 / 3.12	11 / 3.30	11 / 3.32	11 / 3.32
<i>pNDA</i>	10 / 3.61	11 / 4.24	12 / 4.96	12 / 5.11	12 / 5.11
<i>PBJ-ITMM</i>	343 / 100.88	813 / 235.11	N/C	N/C	N/C
<i>P-PI-SI</i>	136 / 54.70	448 / 173.46	N/C	N/C	N/C
<i>AH-PI-SI</i>	182 / 36.21	536 / 103.96	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	494 / 96.08	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	997 / 97.58	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	241 / 47.84	624 / 121.81	N/C	N/C	N/C

Table B.60: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	966 / 103.20	N/C	N/C	N/C	N/C
<i>DSA</i>	11 / 8.25	13 / 11.17	13 / 11.97	13 / 12.65	N/A
<i>AP</i>	10 / 2.76	11 / 3.28	11 / 3.54	11 / 3.50	11 / 3.65
<i>pNDA</i>	10 / 3.72	10 / 4.19	11 / 5.07	11 / 5.03	11 / 5.09
<i>PBJ-ITMM</i>	285 / 90.81	641 / 205.22	N/C	N/C	N/C
<i>P-PI-SI</i>	110 / 44.75	306 / 119.31	N/C	N/C	N/C
<i>AH-PI-SI</i>	148 / 29.70	382 / 74.48	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	495 / 96.26	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	991 / 97.01	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	199 / 39.86	471 / 92.24	N/C	N/C	N/C

Table B.61: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	905 / 96.78	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 4.49	8 / 5.95	8 / 6.68	8 / 6.92	N/A
<i>AP</i>	7 / 1.62	7 / 2.07	7 / 2.34	7 / 2.32	7 / 2.17
<i>pNDA</i>	10 / 2.91	11 / 4.40	11 / 4.82	12 / 5.20	12 / 5.09
<i>PBJ-ITMM</i>	385 / 121.60	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	290 / 113.31	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	661 / 128.09	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	721 / 140.59	N/C	N/C	N/C	N/C

Table B.62: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	952 / 100.51	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 4.26	8 / 5.96	8 / 6.80	8 / 7.19	N/A
<i>AP</i>	8 / 1.80	8 / 2.20	8 / 2.42	8 / 2.50	8 / 2.47
<i>pNDA</i>	10 / 2.78	11 / 4.43	12 / 5.61	12 / 5.46	12 / 5.40
<i>PBJ-ITMM</i>	340 / 109.97	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	252 / 106.20	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	610 / 118.17	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	669 / 130.39	N/C	N/C	N/C	N/C

Table B.63: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	964 / 102.86	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 4.19	8 / 5.42	9 / 7.44	9 / 8.09	N/A
<i>AP</i>	7 / 1.48	7 / 1.98	7 / 2.09	7 / 2.30	7 / 2.43
<i>pNDA</i>	10 / 2.82	10 / 3.26	10 / 4.30	10 / 4.39	10 / 4.35
<i>PBJ-ITMM</i>	315 / 99.55	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	232 / 91.36	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	570 / 110.51	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	628 / 122.41	N/C	N/C	N/C	N/C

Table B.64: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 32, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	978 / 106.80	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 4.05	9 / 5.83	9 / 7.20	9 / 7.52	N/A
<i>AP</i>	7 / 1.52	7 / 1.81	7 / 2.05	7 / 2.15	7 / 2.22
<i>pNDA</i>	10 / 2.59	10 / 3.14	10 / 3.79	10 / 4.18	10 / 4.37
<i>PBJ-ITMM</i>	295 / 96.75	848 / 265.14	N/C	N/C	N/C
<i>P-PI-SI</i>	217 / 85.66	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	536 / 104.03	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	593 / 116.24	N/C	N/C	N/C	N/C

Table B.65: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	864 / 93.75	N/C	N/C	N/C	N/C
<i>DSA</i>	18 / 24.84	20 / 27.14	21 / 30.18	21 / 31.64	N/A
<i>AP</i>	15 / 6.40	17 / 7.28	17 / 7.57	18 / 8.38	18 / 8.04
<i>pNDA</i>	16 / 8.42	18 / 10.12	19 / 11.35	19 / 11.85	19 / 11.41
<i>PBJ-ITMM</i>	260 / 82.21	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	210 / 91.03	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	259 / 55.93	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	494 / 105.79	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	259 / 55.91	N/C	N/C	N/C	N/C

Table B.66: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	916 / 97.44	N/C	N/C	N/C	N/C
<i>DSA</i>	18 / 24.08	20 / 27.28	21 / 30.32	21 / 30.63	N/A
<i>AP</i>	13 / 5.64	15 / 6.87	16 / 7.08	16 / 7.25	16 / 7.36
<i>pNDA</i>	13 / 7.18	15 / 8.54	15 / 9.22	15 / 9.31	15 / 9.40
<i>PBJ-ITMM</i>	134 / 46.32	766 / 238.92	N/C	N/C	N/C
<i>P-PI-SI</i>	102 / 41.65	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	125 / 25.29	765 / 147.44	N/C	N/C	N/C
<i>IPBJ</i>	1000 / 99.40	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	485 / 94.30	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	999 / 97.38	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	125 / 25.47	765 / 148.47	N/C	N/C	N/C

Table B.67: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	919 / 100.83	N/C	N/C	N/C	N/C
<i>DSA</i>	17 / 23.11	20 / 27.04	21 / 30.26	21 / 30.79	N/A
<i>AP</i>	13 / 5.79	15 / 6.80	16 / 7.27	17 / 7.37	17 / 7.88
<i>pNDA</i>	10 / 5.72	12 / 6.68	12 / 7.13	12 / 7.53	12 / 7.43
<i>PBJ-ITMM</i>	129 / 44.35	470 / 152.26	N/C	N/C	N/C
<i>P-PI-SI</i>	57 / 24.44	356 / 138.41	N/C	N/C	N/C
<i>AH-PI-SI</i>	71 / 14.97	469 / 91.14	N/C	N/C	N/C
<i>IPBJ</i>	962 / 100.15	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	473 / 91.98	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	962 / 93.91	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	72 / 15.28	469 / 91.62	N/C	N/C	N/C

Table B.68: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.001$ ,  $\Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	925 / 97.58	N/C	N/C	N/C	N/C
<i>DSA</i>	16 / 20.83	19 / 24.52	21 / 27.77	21 / 28.59	N/A
<i>AP</i>	12 / 4.98	15 / 6.31	17 / 7.43	17 / 7.73	17 / 8.07
<i>pNDA</i>	9 / 5.09	9 / 5.08	10 / 5.85	10 / 5.98	10 / 6.30
<i>PBJ-ITMM</i>	125 / 43.15	270 / 90.08	N/C	N/C	N/C
<i>P-PI-SI</i>	32 / 16.87	193 / 84.19	N/C	N/C	N/C
<i>AH-PI-SI</i>	42 / 10.05	270 / 56.74	N/C	N/C	N/C
<i>IPBJ</i>	949 / 103.61	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	470 / 101.34	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	947 / 101.87	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	44 / 9.83	270 / 53.32	N/C	N/C	N/C

Table B.69: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	865 / 91.01	N/C	N/C	N/C	N/C
<i>DSA</i>	16 / 17.67	17 / 19.11	18 / 20.28	18 / 21.95	N/A
<i>AP</i>	14 / 5.30	15 / 6.25	15 / 6.15	16 / 6.77	16 / 7.00
<i>pNDA</i>	15 / 7.85	17 / 8.23	17 / 8.35	17 / 8.56	17 / 8.49
<i>PBJ-ITMM</i>	264 / 78.18	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	210 / 82.77	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	259 / 50.82	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	495 / 96.23	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	260 / 51.68	N/C	N/C	N/C	N/C

Table B.70: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	916 / 98.22	N/C	N/C	N/C	N/C
<i>DSA</i>	16 / 18.65	18 / 21.17	18 / 21.97	18 / 21.21	N/A
<i>AP</i>	13 / 4.94	14 / 5.47	15 / 5.99	15 / 6.36	15 / 6.40
<i>pNDA</i>	12 / 5.87	14 / 6.92	15 / 8.09	15 / 7.97	15 / 7.82
<i>PBJ-ITMM</i>	203 / 65.71	768 / 242.95	N/C	N/C	N/C
<i>P-PI-SI</i>	102 / 45.58	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	125 / 28.02	767 / 164.95	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	485 / 104.85	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	1000 / 109.48	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	126 / 29.00	767 / 169.27	N/C	N/C	N/C

Table B.71: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ ,  $\Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	921 / 88.29	N/C	N/C	N/C	N/C
<i>DSA</i>	16 / 16.72	18 / 18.81	18 / 19.31	19 / 20.66	N/A
<i>AP</i>	12 / 4.25	14 / 4.93	15 / 5.72	15 / 5.68	15 / 5.82
<i>pNDA</i>	10 / 4.61	12 / 5.60	13 / 6.18	13 / 6.43	13 / 6.37
<i>PBJ-ITMM</i>	193 / 58.23	471 / 137.40	N/C	N/C	N/C
<i>P-PI-SI</i>	57 / 24.50	357 / 139.48	N/C	N/C	N/C
<i>AH-PI-SI</i>	72 / 15.11	470 / 91.08	N/C	N/C	N/C
<i>IPBJ</i>	965 / 95.92	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	474 / 92.20	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	963 / 94.08	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	77 / 16.25	470 / 91.80	N/C	N/C	N/C

Table B.72: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.01$ ,  $\Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	929 / 96.89	N/C	N/C	N/C	N/C
<i>DSA</i>	15 / 17.01	17 / 18.76	18 / 20.70	19 / 22.72	N/A
<i>AP</i>	12 / 4.43	14 / 5.20	15 / 5.70	16 / 6.49	16 / 6.36
<i>pNDA</i>	9 / 4.14	10 / 4.67	11 / 5.63	12 / 6.21	12 / 6.48
<i>PBJ-ITMM</i>	184 / 58.23	273 / 86.24	N/C	N/C	N/C
<i>P-PI-SI</i>	33 / 16.72	194 / 83.08	N/C	N/C	N/C
<i>AH-PI-SI</i>	43 / 9.60	270 / 52.96	N/C	N/C	N/C
<i>IPBJ</i>	955 / 101.93	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	472 / 91.80	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	951 / 92.77	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	66 / 14.30	271 / 53.51	N/C	N/C	N/C

Table B.73: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	880 / 92.84	N/C	N/C	N/C	N/C
<i>DSA</i>	11 / 9.19	12 / 11.01	12 / 10.54	12 / 10.29	N/A
<i>AP</i>	10 / 2.96	10 / 3.20	10 / 3.17	10 / 3.21	10 / 3.16
<i>pNDA</i>	12 / 4.20	13 / 5.22	13 / 5.51	13 / 5.48	13 / 5.68
<i>PBJ-ITMM</i>	485 / 151.41	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	222 / 95.10	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	283 / 55.42	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	350 / 68.74	N/C	N/C	N/C	N/C

Table B.74: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.1$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	926 / 96.82	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 10.06	12 / 10.75	13 / 11.24	13 / 11.71	N/A
<i>AP</i>	10 / 3.04	10 / 3.15	11 / 3.45	11 / 3.62	11 / 3.47
<i>pNDA</i>	11 / 4.30	12 / 4.88	12 / 5.27	12 / 5.28	12 / 5.67
<i>PBJ-ITMM</i>	448 / 138.39	831 / 254.77	N/C	N/C	N/C
<i>P-PI-SI</i>	170 / 74.91	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	221 / 46.15	790 / 161.74	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	492 / 105.88	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	312 / 66.58	799 / 155.12	N/C	N/C	N/C

Table B.75: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	941 / 100.25	N/C	N/C	N/C	N/C
<i>DSA</i>	12 / 9.69	13 / 10.65	13 / 11.11	13 / 11.27	N/A
<i>AP</i>	10 / 2.90	11 / 3.40	11 / 3.81	11 / 3.64	11 / 3.59
<i>pNDA</i>	10 / 3.66	11 / 4.39	11 / 4.56	12 / 5.25	12 / 5.24
<i>PBJ-ITMM</i>	419 / 133.17	656 / 205.24	N/C	N/C	N/C
<i>P-PI-SI</i>	154 / 66.78	379 / 154.05	N/C	N/C	N/C
<i>AH-PI-SI</i>	201 / 43.78	487 / 105.49	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	488 / 102.09	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	985 / 103.07	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	290 / 61.77	514 / 108.59	N/C	N/C	N/C

Table B.76: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.1, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	962 / 92.19	N/C	N/C	N/C	N/C
<i>DSA</i>	11 / 8.19	13 / 10.04	13 / 10.25	13 / 11.00	N/A
<i>AP</i>	9 / 2.43	11 / 3.08	11 / 3.29	11 / 3.27	11 / 3.28
<i>pNDA</i>	9 / 3.03	10 / 3.54	11 / 4.26	11 / 4.28	11 / 4.44
<i>PBJ-ITMM</i>	392 / 114.73	590 / 171.47	N/C	N/C	N/C
<i>P-PI-SI</i>	142 / 57.01	256 / 100.36	N/C	N/C	N/C
<i>AH-PI-SI</i>	185 / 36.86	320 / 62.45	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	493 / 95.74	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	986 / 96.23	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	271 / 53.49	421 / 82.31	N/C	N/C	N/C

Table B.77: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 1$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	911 / 96.59	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 4.75	8 / 6.64	8 / 6.94	8 / 6.92	N/A
<i>AP</i>	7 / 1.65	7 / 1.97	7 / 2.26	7 / 2.14	7 / 2.24
<i>pNDA</i>	10 / 3.18	11 / 4.44	12 / 5.38	12 / 5.22	12 / 5.07
<i>PBJ-ITMM</i>	418 / 136.18	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	308 / 130.16	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	747 / 143.99	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	830 / 160.90	N/C	N/C	N/C	N/C

Table B.78: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64$ ,  $\Sigma_{thin}\Delta u_{thin} = 0.5$ ,  $\Sigma_{thick}\Delta u_{thick} = 2.5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	951 / 99.60	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 4.57	8 / 5.72	8 / 7.04	8 / 7.05	N/A
<i>AP</i>	8 / 1.95	8 / 2.30	8 / 2.44	8 / 2.41	8 / 2.38
<i>pNDA</i>	10 / 2.93	11 / 4.27	N/C	N/C	12 / 4.94
<i>PBJ-ITMM</i>	406 / 127.50	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	298 / 125.88	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	734 / 153.12	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	817 / 158.77	N/C	N/C	N/C	N/C

Table B.79: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 5$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	975 / 98.66	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 3.96	8 / 5.19	9 / 6.97	9 / 7.23	N/A
<i>AP</i>	7 / 1.47	7 / 1.68	7 / 1.92	7 / 1.95	7 / 1.96
<i>pNDA</i>	10 / 2.41	10 / 3.12	10 / 3.62	10 / 3.88	10 / 3.85
<i>PBJ-ITMM</i>	400 / 120.08	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	293 / 114.43	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	725 / 139.79	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	808 / 156.83	N/C	N/C	N/C	N/C

Table B.80: Iterations required (left) and runtime observed (right) to converge parametric study problem with extreme scattering ratios:  $N_s = 64, \Sigma_{thin}\Delta u_{thin} = 0.5, \Sigma_{thick}\Delta u_{thick} = 10$

$c =$	0.99	0.999	0.9999	0.99999	1.0
<i>SI</i>	999 / 107.02	N/C	N/C	N/C	N/C
<i>DSA</i>	8 / 4.39	9 / 6.52	9 / 6.97	9 / 7.83	N/A
<i>AP</i>	7 / 1.58	7 / 1.74	7 / 2.00	7 / 2.09	7 / 2.11
<i>pNDA</i>	10 / 2.71	10 / 3.21	10 / 3.90	10 / 4.25	10 / 4.29
<i>PBJ-ITMM</i>	396 / 122.28	N/C	N/C	N/C	N/C
<i>P-PI-SI</i>	290 / 119.75	N/C	N/C	N/C	N/C
<i>AH-PI-SI</i>	718 / 138.41	N/C	N/C	N/C	N/C
<i>IPBJ</i>	N/C	N/C	N/C	N/C	N/C
<i>P-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-IP-SI</i>	N/C	N/C	N/C	N/C	N/C
<i>AH-PI-IP</i>	801 / 155.67	N/C	N/C	N/C	N/C

## **Appendix C: Full Results from THOR Godiva Testing Suite in Table 5.2**

This appendix contains the number of required iterations (outer and inner) as well as the solution execution times (along with the three constituent times: pre-process, computation, and communication) for each case of the THOR Godiva testing suite in Table 5.2. In the following tables, the number of processors is not explicitly stated, as the data points are iteration or timing data instead. The number of processors for a given case can be obtained by dividing the target mesh size by the number of cells per sub-domain, or by referring to Table 5.2. Selections from this data are plotted as strong and weak scaling trends in Sec. 5.6.1.

Table C.1: Inner iterations consumed by THOR, executing *PBJ-ITMM*, for Godiva testing suite  
*PBJ-ITMM* Inner Iterations

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	216
16 ( $2^4$ )	-	-	-	-	-	-	300	360
32 ( $2^5$ )	-	-	-	-	-	300	348	372
64 ( $2^6$ )	-	-	-	-	312	372	384	444
128 ( $2^7$ )	-	-	-	336	408	408	504	540
256 ( $2^8$ )	-	-	324	384	372	420	516	612
512 ( $2^9$ )	-	348	372	360	444	504	696	804
1024 ( $2^{10}$ )	360	384	420	456	492	564	852	936
2048 ( $2^{11}$ )	420	408	420	492	576	696	876	1188
4096 ( $2^{12}$ )	408	420	528	600	732	876	1080	1320
8192 ( $2^{13}$ )	444	528	624	732	876	1068	1332	1644
16,384 ( $2^{14}$ )	516	636	744	888	1080	1332	1644	2004
32,768 ( $2^{15}$ )	636	756	900	1080	1332	1980	2532	2760
65,536 ( $2^{16}$ )	744	924	1104	1332	1620	2400	3108	3396
131,072 ( $2^{17}$ )	924	1128	1344	1632	1980	2532	3168	3840
262,144 ( $2^{18}$ )	1140	1368	1668	2004	2448	3648	4668	5088
524,288 ( $2^{19}$ )	1392	1680	2028	2460	2976	3636	4620	-
1,048,576 ( $2^{20}$ )	1704	2040	2484	3000	3612	4440	-	-
2,097,152 ( $2^{21}$ )	2076	2508	3048	3672	4440	-	-	-
4,194,304 ( $2^{22}$ )	2532	3072	3720	4488	-	-	-	-
8,388,608 ( $2^{23}$ )	3108	3756	4536	-	-	-	-	-

Table C.2: Inner iterations consumed by THOR, executing *IPBJ*, for Godiva testing suite  
*IPBJ* Inner Iterations

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	396
16 ( $2^4$ )	-	-	-	-	-	-	468	564
32 ( $2^5$ )	-	-	-	-	-	468	528	600
64 ( $2^6$ )	-	-	-	-	492	564	576	672
128 ( $2^7$ )	-	-	-	528	600	612	732	780
256 ( $2^8$ )	-	-	528	588	600	648	768	864
512 ( $2^9$ )	-	540	588	624	684	768	960	1056
1024 ( $2^{10}$ )	552	600	648	708	756	828	1104	1188
2048 ( $2^{11}$ )	636	648	696	756	840	960	1128	1440
4096 ( $2^{12}$ )	648	696	792	876	996	1140	1332	1572
8192 ( $2^{13}$ )	708	804	888	996	1140	1320	1584	1896
16,384 ( $2^{14}$ )	792	912	1020	1140	1332	1584	1896	2244
32,768 ( $2^{15}$ )	900	1020	1164	1344	1584	2232	2784	3012
65,536 ( $2^{16}$ )	1020	1188	1356	1596	1872	2664	3372	3660
131,072 ( $2^{17}$ )	1188	1380	1596	1884	2232	2784	3444	4116
262,144 ( $2^{18}$ )	1392	1632	1920	2256	2700	3924	4980	5412
524,288 ( $2^{19}$ )	1644	1932	2280	2724	3240	3912	4920	-
1,048,576 ( $2^{20}$ )	1956	2292	2736	3276	3900	4740	-	-
2,097,152 ( $2^{21}$ )	2328	2772	3312	3948	4740	-	-	-
4,194,304 ( $2^{22}$ )	2796	3348	3996	4800	-	-	-	-
8,388,608 ( $2^{23}$ )	3372	4044	4848	-	-	-	-	-

Table C.3: Outer iterations consumed by THOR, executing *PBJ-ITMM*, for Godiva testing suite  
*PBJ-ITMM* Outer Iterations

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	18
16 ( $2^4$ )	-	-	-	-	-	-	25	30
32 ( $2^5$ )	-	-	-	-	-	25	29	31
64 ( $2^6$ )	-	-	-	-	26	31	32	37
128 ( $2^7$ )	-	-	-	28	34	34	42	45
256 ( $2^8$ )	-	-	27	32	31	35	43	51
512 ( $2^9$ )	-	29	31	30	37	42	58	67
1024 ( $2^{10}$ )	30	32	35	38	41	47	71	78
2048 ( $2^{11}$ )	35	34	35	41	48	58	73	99
4096 ( $2^{12}$ )	34	35	44	50	61	73	90	110
8192 ( $2^{13}$ )	37	44	52	61	73	89	111	137
16,384 ( $2^{14}$ )	43	53	62	74	90	111	137	167
32,768 ( $2^{15}$ )	53	63	75	90	111	165	211	230
65,536 ( $2^{16}$ )	62	77	92	111	135	200	259	283
131,072 ( $2^{17}$ )	77	94	112	136	165	211	264	320
262,144 ( $2^{18}$ )	95	114	139	167	204	304	389	424
524,288 ( $2^{19}$ )	116	140	169	205	248	303	385	-
1,048,576 ( $2^{20}$ )	142	170	207	250	301	370	-	-
2,097,152 ( $2^{21}$ )	173	209	254	306	370	-	-	-
4,194,304 ( $2^{22}$ )	211	256	310	374	-	-	-	-
8,388,608 ( $2^{23}$ )	259	313	378	-	-	-	-	-

Table C.4: Outer iterations consumed by THOR, executing *IPBJ*, for Godiva testing suite  
*IPBJ* Outer Iterations

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	33
16 ( $2^4$ )	-	-	-	-	-	-	39	47
32 ( $2^5$ )	-	-	-	-	-	39	44	50
64 ( $2^6$ )	-	-	-	-	41	47	48	56
128 ( $2^7$ )	-	-	-	44	50	51	61	65
256 ( $2^8$ )	-	-	44	49	50	54	64	72
512 ( $2^9$ )	-	45	49	52	57	64	80	88
1024 ( $2^{10}$ )	46	50	54	59	63	69	92	99
2048 ( $2^{11}$ )	53	54	58	63	70	80	94	120
4096 ( $2^{12}$ )	54	58	66	73	83	95	111	131
8192 ( $2^{13}$ )	59	67	74	83	95	110	132	158
16,384 ( $2^{14}$ )	66	76	85	95	111	132	158	187
32,768 ( $2^{15}$ )	75	85	97	112	132	186	232	251
65,536 ( $2^{16}$ )	85	99	113	133	156	222	281	305
131,072 ( $2^{17}$ )	99	115	133	157	186	232	287	343
262,144 ( $2^{18}$ )	116	136	160	188	225	327	415	451
524,288 ( $2^{19}$ )	137	161	190	227	270	326	410	-
1,048,576 ( $2^{20}$ )	163	191	228	273	325	395	-	-
2,097,152 ( $2^{21}$ )	194	231	276	329	395	-	-	-
4,194,304 ( $2^{22}$ )	233	279	333	400	-	-	-	-
8,388,608 ( $2^{23}$ )	281	337	404	-	-	-	-	-

Table C.5: Solution time (s) consumed by THOR, executing *PBJ-ITMM*, for Godiva testing suite  
*PBJ-ITMM* Solution Time

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	0.01
16 ( $2^4$ )	-	-	-	-	-	-	0.09	0.04
32 ( $2^5$ )	-	-	-	-	-	0.28	0.09	0.03
64 ( $2^6$ )	-	-	-	-	0.86	0.23	0.07	0.03
128 ( $2^7$ )	-	-	-	3.87	1.1	0.28	0.1	0.04
256 ( $2^8$ )	-	-	14.5	3.99	1.1	0.29	0.11	0.07
512 ( $2^9$ )	-	50.93	13.69	3.8	1.06	0.26	0.13	0.08
1024 ( $2^{10}$ )	189.35	51.34	13.94	3.75	1.03	0.27	0.16	0.1
2048 ( $2^{11}$ )	235.5	61.99	16.51	4.69	1.28	0.41	0.18	0.17
4096 ( $2^{12}$ )	200.48	53.53	15.81	4.46	1.27	0.38	0.22	0.17
8192 ( $2^{13}$ )	205.53	57.47	16.78	4.92	1.32	0.46	0.27	0.55
16,384 ( $2^{14}$ )	210.69	62.3	17.73	5.61	1.44	0.51	0.44	0.86
32,768 ( $2^{15}$ )	227.89	67.87	18.37	5.61	1.59	0.9	0.94	1.31
65,536 ( $2^{16}$ )	239.16	67.89	20.12	6.64	2.18	1.62	2.27	6.68
131,072 ( $2^{17}$ )	266.46	78.43	23.72	7.73	2.95	1.89	5.98	38.3
262,144 ( $2^{18}$ )	287.29	85.61	27.05	10.77	4.58	12.9	43.7	102.48
524,288 ( $2^{19}$ )	310.11	98.32	36.56	15.4	13.03	39.11	122.86	-
1,048,576 ( $2^{20}$ )	350.96	132.2	47.78	39.56	60.66	130.83	-	-
2,097,152 ( $2^{21}$ )	427.56	158.77	89.56	118.6	121.47	-	-	-
4,194,304 ( $2^{22}$ )	494.36	230.24	147.66	156.29	-	-	-	-
8,388,608 ( $2^{23}$ )	796.98	421.8	246.85	-	-	-	-	-

Table C.6: Solution time (s) consumed by THOR, executing *IPBJ*, for Godiva testing suite  
*IPBJ* Solution Time

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	0.07
16 ( $2^4$ )	-	-	-	-	-	-	0.23	0.16
32 ( $2^5$ )	-	-	-	-	-	0.41	0.24	0.15
64 ( $2^6$ )	-	-	-	-	0.74	0.43	0.23	0.15
128 ( $2^7$ )	-	-	-	1.77	1.0	0.53	0.34	0.19
256 ( $2^8$ )	-	-	3.59	1.96	1.01	0.55	0.35	0.21
512 ( $2^9$ )	-	7.2	3.79	2.0	1.09	0.61	0.46	0.31
1024 ( $2^{10}$ )	15.2	7.77	4.04	2.21	1.18	0.7	0.55	0.39
2048 ( $2^{11}$ )	19.18	9.18	4.8	2.53	1.45	0.93	0.62	0.51
4096 ( $2^{12}$ )	17.66	8.89	4.97	2.72	1.65	0.99	0.75	0.52
8192 ( $2^{13}$ )	19.48	10.09	5.49	3.31	1.86	1.33	0.91	0.78
16,384 ( $2^{14}$ )	20.95	11.58	6.55	3.56	2.56	1.43	1.77	1.43
32,768 ( $2^{15}$ )	23.58	12.96	7.17	4.63	3.06	2.31	2.43	3.99
65,536 ( $2^{16}$ )	26.75	15.11	8.65	5.26	3.89	4.12	5.9	7.52
131,072 ( $2^{17}$ )	31.47	18.23	10.98	7.51	7.29	9.25	12.39	45.41
262,144 ( $2^{18}$ )	36.85	21.91	14.87	10.28	11.59	25.01	61.61	148.85
524,288 ( $2^{19}$ )	43.85	29.58	20.87	22.04	23.18	109.51	130.1	-
1,048,576 ( $2^{20}$ )	59.54	39.8	37.37	89.2	71.57	133.06	-	-
2,097,152 ( $2^{21}$ )	81.41	68.98	79.11	144.66	148.04	-	-	-
4,194,304 ( $2^{22}$ )	117.29	113.38	144.24	192.01	-	-	-	-
8,388,608 ( $2^{23}$ )	191.9	215.3	265.01	-	-	-	-	-

Table C.7: Pre-process time (s) consumed by THOR, executing *PBJ-ITMM*, for Godiva testing suite

Target Mesh Size	<i>PBJ-ITMM</i> Pre-Process Time							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	0.01
16 ( $2^4$ )	-	-	-	-	-	-	0.09	0.03
32 ( $2^5$ )	-	-	-	-	-	0.27	0.08	0.03
64 ( $2^6$ )	-	-	-	-	0.77	0.22	0.06	0.02
128 ( $2^7$ )	-	-	-	3.52	0.96	0.27	0.09	0.02
256 ( $2^8$ )	-	-	13.24	3.54	0.97	0.27	0.09	0.03
512 ( $2^9$ )	-	46.08	12.15	3.37	0.87	0.24	0.08	0.02
1024 ( $2^{10}$ )	171.22	45.37	12.15	3.2	0.84	0.24	0.07	0.02
2048 ( $2^{11}$ )	207.66	53.91	14.15	3.74	1.0	0.28	0.08	0.05
4096 ( $2^{12}$ )	173.66	44.25	11.79	3.17	0.89	0.23	0.09	0.04
8192 ( $2^{13}$ )	171.23	44.59	11.55	3.18	0.85	0.24	0.08	0.02
16,384 ( $2^{14}$ )	170.26	45.31	11.78	3.08	0.81	0.24	0.07	0.02
32,768 ( $2^{15}$ )	173.05	44.44	11.31	3.0	0.82	0.24	0.08	0.02
65,536 ( $2^{16}$ )	163.6	43.41	11.29	3.12	0.85	0.24	0.08	0.02
131,072 ( $2^{17}$ )	174.83	44.78	11.79	3.11	0.87	0.23	0.07	0.02
262,144 ( $2^{18}$ )	167.51	43.54	11.28	3.01	0.8	0.25	0.08	0.63
524,288 ( $2^{19}$ )	166.34	43.37	11.12	2.96	0.81	0.23	0.07	-
1,048,576 ( $2^{20}$ )	168.52	43.77	11.58	3.03	0.79	0.23	-	-
2,097,152 ( $2^{21}$ )	183.15	48.95	12.47	3.3	0.98	-	-	-
4,194,304 ( $2^{22}$ )	169.99	43.96	11.51	3.1	-	-	-	-
8,388,608 ( $2^{23}$ )	170.14	43.98	11.92	-	-	-	-	-

Table C.8: Pre-process time (s) consumed by THOR, executing *IPBJ*, for Godiva testing suite  
*IPBJ* Pre-Process Time

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	0.001
16 ( $2^4$ )	-	-	-	-	-	-	0.004	0.002
32 ( $2^5$ )	-	-	-	-	-	0.009	0.004	0.003
64 ( $2^6$ )	-	-	-	-	0.022	0.009	0.004	0.002
128 ( $2^7$ )	-	-	-	0.085	0.03	0.01	0.005	0.002
256 ( $2^8$ )	-	-	0.271	0.096	0.031	0.011	0.006	0.003
512 ( $2^9$ )	-	0.797	0.265	0.091	0.029	0.01	0.006	0.002
1024 ( $2^{10}$ )	2.551	0.843	0.28	0.089	0.03	0.011	0.005	0.004
2048 ( $2^{11}$ )	3.343	1.032	0.312	0.108	0.032	0.018	0.02	0.017
4096 ( $2^{12}$ )	2.882	0.815	0.271	0.085	0.036	0.01	0.037	0.022
8192 ( $2^{13}$ )	3.06	0.862	0.262	0.092	0.04	0.029	0.031	0.007
16,384 ( $2^{14}$ )	2.719	0.864	0.274	0.092	0.03	0.027	0.016	0.025
32,768 ( $2^{15}$ )	2.716	0.937	0.305	0.095	0.052	0.014	0.022	0.074
65,536 ( $2^{16}$ )	2.861	0.86	0.27	0.102	0.03	0.019	0.018	0.002
131,072 ( $2^{17}$ )	2.98	0.974	0.294	0.105	0.04	0.012	0.234	0.003
262,144 ( $2^{18}$ )	3.285	0.862	0.286	0.088	0.31	0.068	0.006	0.014
524,288 ( $2^{19}$ )	2.593	0.817	0.277	0.08	0.029	0.012	0.005	-
1,048,576 ( $2^{20}$ )	2.75	0.842	0.303	0.082	0.028	0.016	-	-
2,097,152 ( $2^{21}$ )	2.801	1.225	0.708	0.101	0.026	-	-	-
4,194,304 ( $2^{22}$ )	2.601	0.876	0.292	0.082	-	-	-	-
8,388,608 ( $2^{23}$ )	2.682	0.827	0.255	-	-	-	-	-

Table C.9: Computation time (s) consumed by THOR, executing *PBJ-ITMM*, for Godiva testing suite

Target Mesh Size	<i>PBJ-ITMM</i> Computation Time							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	<0.01
16 ( $2^4$ )	-	-	-	-	-	-	0.01	<0.01
32 ( $2^5$ )	-	-	-	-	-	0.01	0.01	<0.01
64 ( $2^6$ )	-	-	-	-	0.08	0.01	0.01	<0.01
128 ( $2^7$ )	-	-	-	0.35	0.13	0.02	0.01	0.01
256 ( $2^8$ )	-	-	1.25	0.45	0.12	0.02	0.01	0.03
512 ( $2^9$ )	-	4.85	1.54	0.41	0.16	0.02	0.04	0.04
1024 ( $2^{10}$ )	18.12	5.95	1.43	0.5	0.16	0.02	0.05	0.05
2048 ( $2^{11}$ )	26.12	7.22	2.2	0.77	0.25	0.09	0.07	0.08
4096 ( $2^{12}$ )	20.47	7.24	2.56	1.07	0.31	0.11	0.08	0.08
8192 ( $2^{13}$ )	22.4	9.15	3.83	1.42	0.29	0.14	0.11	0.13
16,384 ( $2^{14}$ )	29.73	14.5	5	1.51	0.32	0.18	0.15	0.16
32,768 ( $2^{15}$ )	47.84	15.51	4.83	1.76	0.52	0.23	0.26	0.24
65,536 ( $2^{16}$ )	49.05	18.01	6.36	2.2	0.68	0.33	0.3	0.31
131,072 ( $2^{17}$ )	60.42	22.65	8.32	2.93	0.95	0.39	0.3	1.09
262,144 ( $2^{18}$ )	82.27	29.88	9.6	3.59	0.92	0.57	1.27	0.28
524,288 ( $2^{19}$ )	114.27	29.97	12.39	4.18	1.42	0.49	1.39	-
1,048,576 ( $2^{20}$ )	105.33	41.46	15.97	5.11	1.67	0.48	-	-
2,097,152 ( $2^{21}$ )	152.63	52.17	18.9	6.57	2.36	-	-	-
4,194,304 ( $2^{22}$ )	170.27	63.99	21.55	7.81	-	-	-	-
8,388,608 ( $2^{23}$ )	208.13	72.94	26.37	-	-	-	-	-

Table C.10: Computation time (s) consumed by THOR, executing *IPBJ*, for Godiva testing suite  
*IPBJ* Computation Time

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	0.07
16 ( $2^4$ )	-	-	-	-	-	-	0.23	0.15
32 ( $2^5$ )	-	-	-	-	-	0.4	0.23	0.13
64 ( $2^6$ )	-	-	-	-	0.72	0.42	0.21	0.13
128 ( $2^7$ )	-	-	-	1.68	0.96	0.51	0.3	0.17
256 ( $2^8$ )	-	-	3.32	1.82	0.94	0.54	0.32	0.18
512 ( $2^9$ )	-	6.39	3.48	1.9	1.04	0.59	0.44	0.28
1024 ( $2^{10}$ )	12.64	6.91	3.63	2.06	1.11	0.68	0.51	0.33
2048 ( $2^{11}$ )	15.72	7.89	4.37	2.31	1.39	0.86	0.56	0.44
4096 ( $2^{12}$ )	14.54	7.85	4.36	2.56	1.55	0.91	0.63	0.41
8192 ( $2^{13}$ )	15.71	8.99	5.01	2.93	1.67	0.97	0.73	0.48
16,384 ( $2^{14}$ )	17.47	10.27	5.73	3.21	1.96	1.2	0.82	0.63
32,768 ( $2^{15}$ )	20.67	11.12	6.28	3.97	2.24	1.81	1.31	0.82
65,536 ( $2^{16}$ )	22.34	13.2	7.38	4.6	2.73	2.11	1.44	0.86
131,072 ( $2^{17}$ )	26.57	15.34	8.82	5.29	3.4	2.15	1.54	1.59
262,144 ( $2^{18}$ )	30.9	17.85	10.94	6.44	4.1	3.07	2.29	1.27
524,288 ( $2^{19}$ )	35.84	21.35	12.28	7.82	4.65	3.04	3.94	-
1,048,576 ( $2^{20}$ )	42.66	24.49	15.52	9.57	5.82	3.71	-	-
2,097,152 ( $2^{21}$ )	54.23	31.43	20.69	11.66	8.29	-	-	-
4,194,304 ( $2^{22}$ )	60.98	36.21	22.67	14.63	-	-	-	-
8,388,608 ( $2^{23}$ )	76.25	44.51	28.03	-	-	-	-	-

Table C.11: Communication time (s) consumed by THOR, executing *PBJ-ITMM*, for Godiva testing suite

Target Mesh Size	<i>PBJ-ITMM</i> Communication Time							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	0.0
16 ( $2^4$ )	-	-	-	-	-	-	0.0	<0.01
32 ( $2^5$ )	-	-	-	-	-	0.0	<0.01	<0.01
64 ( $2^6$ )	-	-	-	-	0.0	<0.01	<0.01	<0.01
128 ( $2^7$ )	-	-	-	0.0	<0.01	<0.01	<0.01	0.01
256 ( $2^8$ )	-	-	0.0	<0.01	0.01	<0.01	0.01	0.01
512 ( $2^9$ )	-	0.0	0.01	0.02	0.03	<0.01	0.01	0.02
1024 ( $2^{10}$ )	0.0	0.02	0.36	0.05	0.04	0.01	0.04	0.03
2048 ( $2^{11}$ )	1.72	0.86	0.16	0.18	0.04	0.03	0.03	0.04
4096 ( $2^{12}$ )	6.36	2.04	1.46	0.22	0.07	0.04	0.05	0.05
8192 ( $2^{13}$ )	11.9	3.73	1.4	0.32	0.19	0.08	0.08	0.4
16,384 ( $2^{14}$ )	10.7	2.49	0.96	1.03	0.31	0.09	0.23	0.68
32,768 ( $2^{15}$ )	7.0	7.92	2.22	0.84	0.25	0.43	0.6	1.05
65,536 ( $2^{16}$ )	26.51	6.46	2.47	1.32	0.65	1.05	1.89	6.34
131,072 ( $2^{17}$ )	31.21	11.01	3.61	1.68	1.13	1.26	5.61	37.18
262,144 ( $2^{18}$ )	37.51	12.19	6.17	4.17	2.86	12.08	42.35	101.56
524,288 ( $2^{19}$ )	29.5	24.98	13.04	8.26	10.79	38.39	121.4	-
1,048,576 ( $2^{20}$ )	77.11	46.98	20.23	31.41	58.19	130.11	-	-
2,097,152 ( $2^{21}$ )	91.78	57.65	58.19	108.73	118.12	-	-	-
4,194,304 ( $2^{22}$ )	154.1	122.29	114.61	145.38	-	-	-	-
8,388,608 ( $2^{23}$ )	418.71	304.88	208.56	-	-	-	-	-

Table C.12: Communication time consumed by THOR, executing *IPBJ*, for Godiva testing suite  
*IPBJ* Communication Time

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
8 ( $2^3$ )	-	-	-	-	-	-	-	0.0
16 ( $2^4$ )	-	-	-	-	-	-	0.0	<0.01
32 ( $2^5$ )	-	-	-	-	-	0.0	<0.01	0.02
64 ( $2^6$ )	-	-	-	-	0.0	<0.01	0.02	0.02
128 ( $2^7$ )	-	-	-	0.0	0.01	0.01	0.03	0.02
256 ( $2^8$ )	-	-	0.0	0.05	0.04	<0.01	0.03	0.03
512 ( $2^9$ )	-	0.0	0.05	0.01	0.02	0.01	0.01	0.03
1024 ( $2^{10}$ )	0.0	0.01	0.13	0.06	0.03	0.02	0.03	0.06
2048 ( $2^{11}$ )	0.12	0.26	0.12	0.12	0.03	0.05	0.04	0.05
4096 ( $2^{12}$ )	0.24	0.23	0.34	0.07	0.06	0.07	0.09	0.09
8192 ( $2^{13}$ )	0.71	0.24	0.22	0.29	0.15	0.33	0.15	0.3
16,384 ( $2^{14}$ )	0.76	0.45	0.54	0.26	0.58	0.2	0.94	0.78
32,768 ( $2^{15}$ )	0.19	0.9	0.58	0.58	0.77	0.49	1.09	3.1
65,536 ( $2^{16}$ )	1.54	1.04	0.99	0.56	1.13	2.0	4.44	6.65
131,072 ( $2^{17}$ )	1.92	1.92	1.87	2.11	3.86	7.09	10.61	43.82
262,144 ( $2^{18}$ )	2.66	3.2	3.64	3.75	7.18	21.87	59.31	147.56
524,288 ( $2^{19}$ )	5.42	7.42	8.32	14.13	18.5	106.46	126.15	-
1,048,576 ( $2^{20}$ )	14.13	14.48	21.54	79.54	65.72	129.34	-	-
2,097,152 ( $2^{21}$ )	24.38	36.32	57.72	132.9	139.72	-	-	-
4,194,304 ( $2^{22}$ )	53.71	76.29	121.27	177.29	-	-	-	-
8,388,608 ( $2^{23}$ )	112.97	169.96	236.73	-	-	-	-	-

## Appendix D: Full Results from THOR C5G7 Testing Suite in Table 5.4

This appendix contains the number of required iterations (outer and inner) as well as the solution execution times (along with the three constituent times: pre-process, computation, and communication) for each case of the THOR C5G7 testing suite in Table 5.4. In the following tables, the number of processors is not explicitly stated, as the data points are iteration or timing data instead. The number of processors for a given case can be obtained by dividing the target mesh size by the number of cells per sub-domain, or by referring to Table 5.4. Selections from this data are plotted as strong and weak scaling trends in Sec. 5.6.2.

Table D.1: Inner iterations consumed by THOR, executing *PBJ-ITMM*, for C5G7 testing suite  
*PBJ-ITMM* Inner Iterations

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	26,866	28,588	31,080	34,342	39,116	46,326	57,232	80,500
524,288 ( $2^{19}$ )	18,242	20,804	24,346	29,120	35,938	45,780	59,990	-
1,048,576 ( $2^{20}$ )	15,638	19,390	24,514	30,590	37,464	46,858	-	-
2,097,152 ( $2^{21}$ )	16,436	20,398	24,962	30,898	37,842	-	-	-
4,194,304 ( $2^{22}$ )	16,870	20,678	25,438	31,052	-	-	-	-
8,388,608 ( $2^{23}$ )	16,646	20,384	25,004	-	-	-	-	-

Table D.2: Inner iterations consumed by THOR, executing *IPBJ*, for C5G7 testing suite  
*IPBJ* Inner Iterations

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	33,810	35,350	37,688	40,614	45,150	51,884	62,202	84,756
524,288 ( $2^{19}$ )	26,124	28,406	31,514	35,742	41,916	51,226	65,058	-
1,048,576 ( $2^{20}$ )	24,010	27,202	31,626	37,128	43,288	52,192	-	-
2,097,152 ( $2^{21}$ )	25,340	28,700	32,634	37,926	44,212	-	-	-
4,194,304 ( $2^{22}$ )	26,208	29,708	33,880	38,682	-	-	-	-
8,388,608 ( $2^{23}$ )	27,202	30,618	34,636	-	-	-	-	-

Table D.3: Outer iterations consumed by THOR, executing *PBJ-ITMM*, for C5G7 testing suite  
*PBJ-ITMM* Outer Iterations

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	1919	2042	2220	2453	2794	3309	4088	5750
524,288 ( $2^{19}$ )	1303	1486	1739	2080	2567	3270	4285	-
1,048,576 ( $2^{20}$ )	1117	1385	1751	2185	2676	3347	-	-
2,097,152 ( $2^{21}$ )	1174	1457	1783	2207	2703	-	-	-
4,194,304 ( $2^{22}$ )	1205	1477	1817	2218	-	-	-	-
8,388,608 ( $2^{23}$ )	1189	1456	1786	-	-	-	-	-

Table D.4: Outer iterations consumed by THOR, executing *IPBJ*, for C5G7 testing suite  
*IPBJ* Outer Iterations

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	2415	2525	2692	2901	3225	3706	4443	6054
524,288 ( $2^{19}$ )	1866	2029	2251	2553	2994	3659	4647	-
1,048,576 ( $2^{20}$ )	1715	1943	2259	2652	3092	3728	-	-
2,097,152 ( $2^{21}$ )	1810	2050	2331	2709	3158	-	-	-
4,194,304 ( $2^{22}$ )	1872	2122	2420	2763	-	-	-	-
8,388,608 ( $2^{23}$ )	1943	2187	2474	-	-	-	-	-

Table D.5: Solution time (minutes) consumed by THOR, executing *PBJ-ITMM*, for C5G7 testing suite

Target Mesh Size	<i>PBJ-ITMM</i> Solution Time							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	85.4	23.2	7.5	2.7	1.2	0.9	6.0	31.6
524,288 ( $2^{19}$ )	52.3	13.5	5.1	2.7	2.3	5.6	20.0	-
1,048,576 ( $2^{20}$ )	30.3	12.5	6.2	4.9	6.9	16.1	-	-
2,097,152 ( $2^{21}$ )	34.8	15.3	9.1	9.2	14.1	-	-	-
4,194,304 ( $2^{22}$ )	37.1	18.5	17.0	15.2	-	-	-	-
8,388,608 ( $2^{23}$ )	41.0	23.1	19.4	-	-	-	-	-

Table D.6: Solution time (minutes) consumed by THOR, executing *IPBJ*, for C5G7 testing suite

Target Mesh Size	<i>IPBJ</i> Solution Time							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	14.8	8.3	4.9	3.3	4.0	4.1	9.0	35.1
524,288 ( $2^{19}$ )	12.2	7.4	4.9	5.6	5.1	10.1	27.5	-
1,048,576 ( $2^{20}$ )	12.0	7.9	7.3	7.5	10.7	24.4	-	-
2,097,152 ( $2^{21}$ )	14.2	11.1	10.4	13.2	20.1	-	-	-
4,194,304 ( $2^{22}$ )	19.7	16.7	17.8	22.9	-	-	-	-
8,388,608 ( $2^{23}$ )	26.1	23.6	26.9	-	-	-	-	-

Table D.7: Pre-process time (s) consumed by THOR, executing *PBJ-ITMM*, for C5G7 testing suite  
*PBJ-ITMM* Pre-Process Time

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	251.03	64.62	16.43	3.98	1.11	0.30	0.08	0.09
524,288 ( $2^{19}$ )	227.15	56.37	15.01	3.90	1.03	0.29	0.09	-
1,048,576 ( $2^{20}$ )	215.98	57.77	14.68	3.90	1.07	0.29	-	-
2,097,152 ( $2^{21}$ )	217.01	56.44	14.66	3.97	1.04	-	-	-
4,194,304 ( $2^{22}$ )	216.94	57.53	18.82	3.83	-	-	-	-
8,388,608 ( $2^{23}$ )	213.70	56.69	15.16	-	-	-	-	-

Table D.8: Pre-process time (s) consumed by THOR, executing *IPBJ*, for C5G7 testing suite  
*IPBJ* Pre-Process Time

Target Mesh Size	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	5.67	1.58	0.48	0.11	0.04	0.01	0.01	0.33
524,288 ( $2^{19}$ )	3.64	1.01	0.32	0.10	0.03	0.01	0.01	-
1,048,576 ( $2^{20}$ )	2.79	0.94	0.28	0.11	0.04	0.34	-	-
2,097,152 ( $2^{21}$ )	3.13	1.73	0.28	0.11	1.67	-	-	-
4,194,304 ( $2^{22}$ )	2.89	1.11	0.34	0.10	-	-	-	-
8,388,608 ( $2^{23}$ )	3.27	0.97	0.34	-	-	-	-	-

Table D.9: Computation time (s) consumed by THOR, executing *PBJ-ITMM*, for C5G7 testing suite

Target Mesh Size	<i>PBJ-ITMM</i> Computation Time							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	3292.5	1017.7	281.2	83.3	25.6	4.5	8.9	10.1
524,288 ( $2^{19}$ )	1744.4	451.7	179.1	63.0	15.6	6.8	15.8	-
1,048,576 ( $2^{20}$ )	1086.0	453.7	163.0	64.6	26.1	21.4	-	-
2,097,152 ( $2^{21}$ )	1184.0	459.0	172.5	66.5	13.6	-	-	-
4,194,304 ( $2^{22}$ )	1469.7	426.1	170.8	62.0	-	-	-	-
8,388,608 ( $2^{23}$ )	1286.1	458.4	177.3	-	-	-	-	-

Table D.10: Computation time (s) consumed by THOR, executing *IPBJ*, for C5G7 testing suite

Target Mesh Size	<i>IPBJ</i> Computation Time							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	796.0	431.5	227.2	121.0	73.2	47.1	25.5	59.7
524,288 ( $2^{19}$ )	642.0	333.9	187.2	109.4	67.5	40.8	40.5	-
1,048,576 ( $2^{20}$ )	570.8	322.7	189.1	113.6	66.4	68.7	-	-
2,097,152 ( $2^{21}$ )	572.7	335.8	189.8	113.2	79.5	-	-	-
4,194,304 ( $2^{22}$ )	622.5	334.9	203.9	117.1	-	-	-	-
8,388,608 ( $2^{23}$ )	627.4	362.5	209.4	-	-	-	-	-

Table D.11: Communication time (minutes) consumed by THOR, executing *PBJ-ITMM*, for C5G7 testing suite

Target Mesh Size	<i>PBJ-ITMM</i> Communication Time							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	26.4	5.2	2.5	1.2	0.7	0.8	5.9	31.4
524,288 ( $2^{19}$ )	19.4	5.1	1.9	1.6	2.1	5.5	19.8	-
1,048,576 ( $2^{20}$ )	8.6	3.9	3.2	3.8	6.5	15.8	-	-
2,097,152 ( $2^{21}$ )	11.5	6.8	6.0	8.1	13.9	-	-	-
4,194,304 ( $2^{22}$ )	9.0	10.4	13.9	14.1	-	-	-	-
8,388,608 ( $2^{23}$ )	16.0	14.5	16.2	-	-	-	-	-

Table D.12: Communication time (minutes) consumed by THOR, executing *IPBJ*, for C5G7 testing suite

Target Mesh Size	<i>IPBJ</i> Communication Time							
	Cells per Sub-Domain							
	1024	512	256	128	64	32	16	8
262,144 ( $2^{18}$ )	1.5	1.0	1.1	1.2	2.7	3.4	8.6	34.1
524,288 ( $2^{19}$ )	1.5	1.8	1.8	3.7	4.0	9.4	26.8	-
1,048,576 ( $2^{20}$ )	2.4	2.5	4.1	5.6	9.6	23.3	-	-
2,097,152 ( $2^{21}$ )	4.6	5.5	7.3	11.3	18.8	-	-	-
4,194,304 ( $2^{22}$ )	9.3	11.1	14.4	21.0	-	-	-	-
8,388,608 ( $2^{23}$ )	15.6	17.6	23.4	-	-	-	-	-