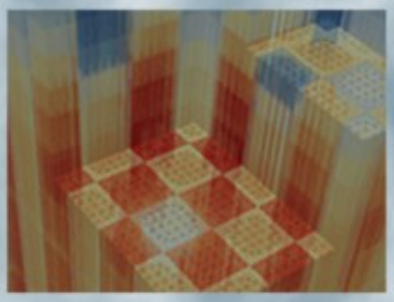
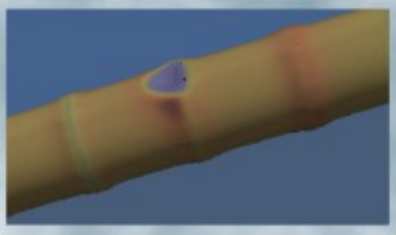
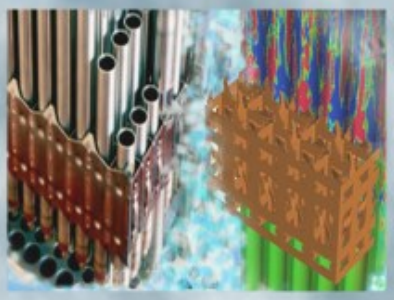
CASL-U-2016-1063-001







MPACT Subgroup Self-Shielding Efficiency Improvements

Shane Stimpson, Oak Ridge National Laboratory
Yuxuan Liu, University of Michigan
Benjamin Collins, Oak Ridge National Laboratory
Kevin Clarno, Oak Ridge National Laboratory

August 31, 2016

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

REVISION LOG

Revision	Date	Affected Pages	Revision Description
0	04/04/2016	All	Original Report
1	08/31/2016	All	Cleared PTS

Document pages that are:

Export Controlled NO

IP/Proprietary/NDA Controlled NO

Sensitive Controlled NO

Approved for Public Release YES

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, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ACRONYMS

MOC	Method of Characteristics
ORNL	Oak Ridge National Laboratory
VERA	Virtual Environment for Reactor Applications

1. INTRODUCTION

Recent developments to improve the efficiency of the MOC solvers in MPACT have yielded effective kernels that loop over several energy groups at once, rather that looping over one group at a time [1]. These kernels have produced roughly a 2x speedup on the MOC sweeping time during eigenvalue calculation. However, the self-shielding subgroup calculation had not been reevaluated to take advantage of these new kernels, which typically requires substantial solve time. The improvements covered in this report start by integrating the multigroup kernel concepts into the subgroup calculation, which are then used as the basis for further extensions.

The next improvement that is covered is what is currently being termed as "Lumped Parameter MOC". Because the subgroup calculation is a purely fixed source problem and multiple sweeps are performed only to update the boundary angular fluxes, the sweep procedure can be condensed to allow for the instantaneous propagation of the flux across a spatial domain, without the need to sweep along all segments in a ray. Once the boundary angular fluxes are considered to be converged, an additional sweep that will tally the scalar flux is completed.

The last improvement that is investigated is the possible reduction of the number of azimuthal angles per octant in the shielding sweep. Typically 16 azimuthal angles per octant are used for self-shielding and eigenvalue calculations, but it is possible that the self-shielding sweeps are less sensitive to the number of angles than the full eigenvalue calculation.

2. BACKGROUND ON IMPROVEMENTS

The general theory behind the subgroup calculation is going to be omitted from this section, only outlining the theory behind the improvements that have been made to it. For a more basic background on subgroup, consult Reference 2.

2.1 Solving All "Pseudogroups" Concurrently

The pre-existing subgroup calculation scheme in MPACT is shown in Figure 1, where there are three loops: 1) over the resonant groups, 2) over the subgroup categories for that group, and 3) over the subgroup levels. As can be seen, inside these loops, there is an iteration loop where a transport sweep for each resonant group, category, and level are performed.

```
for each resonant group (g from g_{res,beg} to g_{res,end})
    for each subgroup category (c from 1 to N_{cat}(g))
        for each subgroup level (l from 1 to N_{levels})
            Setup Σ_{t,g,c,l} based on Σ_{a,g,c,l} and λ_g Σ_p
            Setup source for this group/category/level based on λ_g Σ_p
            for each iteration (i from 1 to N_{iters})
                Perform transport sweep for this group/category/level
                Compare residual based on scalar flux (terminate if below criteria)
            end for
            Calculate equivalence cross section (Σ_{eq,g,c,l})
        end for
    end for
end for
```

Figure 1. Pseudocode for Pre-existing Subgroup Scheme with Group on Outermost Loop

For most MPACT calculations, there are 4 categories per group and 4 levels. But the number of categories can be changed by choosing a different subgroup set, based on the user input [3]. The

number of levels is dependent upon the library being used. Additionally, all results covered in this work make use of the 47-group library developed by ORNL [4], which contains 17 resonant groups (from group 10 to group 26).

To take advantage of the new multigroup kernels that have been implemented into MPACT, the scheme needs to be restructured slightly. To make this concept easier, we are going to say that each resonant group, category, level combination makes up a single "pseudogroup". So the number of pseudogroups for the entire subgroup calculation will be the product of the number of resonant groups times the average number of subgroup categories per group times the number of subgroup levels. In theory, the number of categories can vary from group to group, though this does not seem to be the case for the 47-group library, which yields 272 pseudogroups. Based on this concept, a transport kernel could be constructed to sweep over all pseudogroups concurrently, but the source, cross section, scalar flux, and angular flux need to be stored for each up-front, whereas in the previous scheme, only one group of storage was necessary at a time. Figure 2 shows the pseudocode for the refactored scheme, taking advantage of the multigroup kernel concept:

```
for each pseudogroup (pg from 1 to N_pseudogroups)
    Setup and store Σ_t,pg for this pseudogroup based on Σ_a,pg and λ_pg Σ_p
    Setup and store source for this pseudogroup based on λ_pg Σ_p
end for
for each iteration (i from 1 to N_iters)
    Perform transport sweep for all pseudogroups
    Compare residual based on scalar flux (terminate if below criteria)
end for
for each pseudogroup (pg from 1 to N_pseudogroups)
    Calculate equivalence cross section Σ_eq,pg
end for
```

Figure 2. Pseudocode for Subgroup Scheme Using the Multigroup Transport Kernel

As one might expect, the memory of storing source and flux data for 272 pseudogroups can be a concern. One way of keeping the memory low, while still allowing the scheme to make use the multigroup kernels is to divide the pseudogroups up using batches. Figure 3 shows the pseudocode for the batched approach, where each batch contains a starting and stopping pseudogroup index:

```
for each batch (b from 1 to N_batch)
    for each pseudogroup (pg from pg_beg(b) to pg_end(b))
        Setup and store Σ_t,pg for this pseudogroup based on Σ_a,pg and λ_pg Σ_p
        Setup and store source for this pseudogroup based on λ_pg Σ_p
    end for
    for each iteration (i from 1 to N_iters)
        Perform transport sweep for all pseudogroups
        Compare residual based on scalar flux (terminate if below criteria)
    end for
    for each pseudogroup (pg from pg_beg(b) to pg_end(b))
        Calculate equivalence cross section Σ_eq,pg
    end for
end for
```

Figure 3. Pseudocode for Subgroup Scheme Using the Multigroup Transport Kernel and Batching

The performance of the pseudogroup consolidation alone was not reported. However, based on the results from Ref. 1, which observed a speedup of roughly 1.4x using 47-group kernels without current tallies, it would not be unreasonable to expect a similar performance boost, especially if the number of pseudogroups in a batch is roughly 40-50. The performance of this in addition to the next improvement are the focus of the results in Section 3.

When using the multigroup kernels during the eigenvalue sweep [1], a Jacobi approximation is applied to the scattering source since all groups are solved in the same kernel and the inscatter source is not updated during the kernel sweep. This approximation typically requires more outer iterations over the Gauss-Seidel approach. It is worth noting that when applying these kernels to the subgroup calculation, no additional iterations are required because there is no approximation to the source, though this is not surprising.

2.2 Lumped Parameter MOC

Because the subgroup calculation solves a purely absorbing problem with a fixed source, the transport sweeps can be vastly simplified by condensing the angular flux propagation into one operation, instead of sweeping across all segments along a ray. Because the MOC kernels in MPACT sweep over two angles travelling in opposite directions (forward/backward) at the same time, effectively two equations are needed (Eqs 1a and 1b):

$$\varphi_{pg}^{out,for} = \varphi_{pg}^{in,for} A_{pg} + B_{pg} \tag{1a}$$

$$\varphi_{pg}^{out,back} = \varphi_{pg}^{in,back} A_{pg} + C_{pg} \tag{1b}$$

To visualize this consider a ray in a simple pin cell problem (Figure 4). On the left is the discretization showing 5 segments along the ray (blue) with the incoming and outgoing angular fluxes at the ends of the ray. On the right is the same problem but with all 5 segments condensed into one. To reiterate, this is only valid and effective because the source is not changing between iterations as is the case during the eigenvalue calculation sweeps. Thus, the $A/B/C$ lumped parameters can be used in a fast, intermediate kernel that only updates the outgoing angular flux.

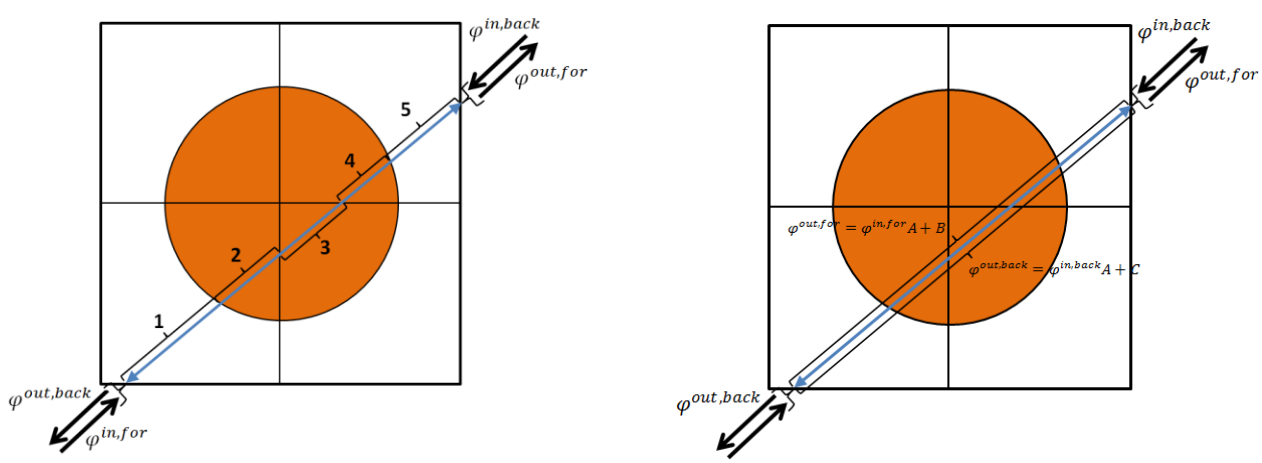


Figure 4. Visualization of MOC Ray Tracing (left) and Lumped Parameter (right) on a Pin Cell

There are a couple of ways of deriving the equations for the lumped parameters. However, it is not difficult to convince yourself that A will be a product of the exponential terms for each segment (Eq. 2a). With A in hand, B and C can be easily calculated using the incoming and outgoing angular flux values (Eqs. 2b and 2c), assuming a typical sweep is performed in calculating the factors. A more

detailed derivation with more explicit formulas for B and C is available in Appendix A, but the implementation in MPACT is consistent with the formulas outlined in Eq. 2.

$$A_{pg} = \prod_{i=1}^{N_{seg}} e^{-\Sigma_{t,i,pg} l_i} \tag{2a}$$

$$B_{pg} = \varphi_{pg}^{out,for} - \varphi_{pg}^{in,for} A_{pg} \tag{2b}$$

$$C_{pg} = \varphi_{pg}^{out,back} - \varphi_{pg}^{in,back} A_{pg} \tag{2c}$$

It is worth noting that the notation in Eqs. 1 and 2 are heavily simplified to avoid clutter, but the lumped parameters will need to be calculated and saved for each angle and ray. However, given that these will only be three values over O(100) segments, the storage for this is not concerning.

Figure 5 shows the pseudocode for lumped parameters, building off of the multigroup kernel with batching. The important changes here to note are that 1) there is an initial sweep to calculate the lumped parameters (line 6), 2) there are several "fast" sweeps that simply apply the factors to update the angular flux (per Eq. 1 and line 8), and 3) a final, standard sweep is completed to tally the scalar flux (line 11), which is required for the equivalence cross section calculation (line 13).

```
1:  for each batch (b from 1 to N_batch)
2:      for each pseudogroup (pg from pg_beg(b) to pg_end(b))
3:          Setup and store Σ_t,pg for this pseudogroup based on Σ_a,pg and λ_pg Σ_p
4:          Setup and store source for this pseudogroup based on λ_pg Σ_p
5:      end for
6:      Perform an initial sweep accumulating the A_pg/B_pg/C_pg lumped parameters
7:      for each iteration (i from 2 to N_iters)
8:          Perform a transport sweep applying A_pg/B_pg/C_pg parameters for this batch
9:          Compare residual based on boundary angular fluxes (terminate if below criteria)
10:     end for
11:     Perform a final, normal sweep accumulating scalar flux (φ_pg)
12:     for each pseudogroup (pg from pg_beg(b) to pg_end(b))
13:         Calculate equivalence cross section Σ_eq,pg
14:     end for
15: end for
```

Figure 5. Pseudocode for Subgroup Scheme Using the Multigroup Transport Kernel, Batching, and Lumped Parameter Approach

Since only the last iteration yields a scalar flux distribution, the convergence residual for this scheme is based on the angular flux updates instead of the scalar flux, which is used in the current scheme. Choosing the correct convergence criteria is important to ensure consistency between these two schemes. The current scheme imposes a maximum change of $1x10^{-6}$ for the scalar flux in any region for each pseudogroup. Since the new scheme will perform an additional sweep once the angular flux is considered to be converged, a similar maximum change is imposed on the angular flux, but with a criteria of $1x10^{-5}$. In practice, this has been observed to be conservative, in most cases requiring one additional iteration. However, this is tolerable since it is only one additional "fast" iteration.

One should also keep in mind a situation where this approach would not be beneficial, which is in problems with fully vacuum radial boundary conditions in serial. In this scenario, only one iteration would be necessary since the boundary conditions do not need to be converged as an zero incoming angular flux is correct. This, however, is not a likely scenario given most problems are executed with quarter symmetry and in parallel.

2.3 Reducing Azimuthal Angles

One additional improvement that will be covered separately in the results section is the notion that fewer azimuthal angles are necessary in the quadrature when performing the subgroup calculation. The self-shielding parameters in the 47-group library were generated using 8 azimuthal angles per octant [4]. In theory, the answers may improve when using only 8 angles, compared to the 16 angles that are default, but it is also suspected that the subgroup results may be more insensitive than for the eigenvalue sweeps.

3. RESULTS

The improvements covered in the previous section have been applied to three tests problems that will be presented here, all from the VERA progression problem suite [5]: 1) a single quarter assembly lattice [VERA Problem 2a, Figure 6] and 2) a 2D slice of the 3x3 assembly cluster [VERA Problem 4a-2D, Figure 7], and 3) a 2D quarter core model [VERA Problem 5a-2D core layout in Figure 8]. It is worth noting that the 5a-2D case also includes baffle and reflector regions that are not included in the figure. All cases used a 0.05 cm ray spacing, 16 azimuthal angles per octant, and 2 polar angles in a Tabuchi-Yamamoto [6] quadrature with 3 radial rings in the fuel and 8 azimuthal divisions. As mentioned before, all cases used a 47-group cross section library generated by ORNL [4]. Both problems 2a and 4a-2D were run in serial on local ORNL clusters with AMD processors. P5-2D was run on Titan [7] using 73 spatial decomposition domains.

In all Tables 1-3, the number of batches used to partition the pseudogroups is varied between 1-10, with the average number of pseudogroups in each batch reported along with the total time spent performing the subgroup calculation (sec) and the total memory for the problem. Additionally, results from the pre-existing one-group sweeping scheme (1G) is reported as well as the memory for the multigroup scheme (MG) allocating 47-groups of data, which is most directly comparable to the case with 6 batches at a maximum of 46 groups per batch. The time for the MG scheme is not reported since it is expected to be very close to the 1G time, since it was originally implemented to use 1G for the self-shielding and MG for the eigenvalue calculation.

3.1 VERA Problem 2a

Figure 6 shows the visualization of the Problem 2a geometry, which is a simple 17x17 lattice of 2.1% enriched pins using quarter symmetry.

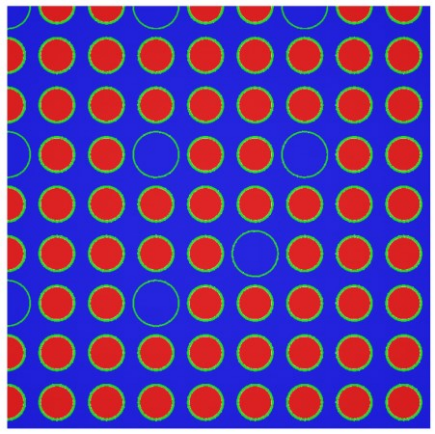


Figure 6. Geometry Visualization of VERA Problem 2a [5]

Table 1 shows the results for Problem 2A. There are several conclusions that can be drawn from this. The first is that batching can significantly reduce the memory burden of the new scheme, requiring only a little more memory than compared to the old scheme, without sacrificing much performance. Another is that the new scheme is significantly faster, providing roughly a 2.8x speedup when using 5 batches, which is the current default.

Table 1. Results for Problem 2a

# Batch	Average Batch Size (pseudogroups)	Subgroup Time (sec)	Total Memory (GB)
1	272.0	8.17	0.50
2	136.0	8.25	0.35
3	90.7	8.26	0.30
4	68.0	8.42	0.27
5	54.4	8.70	0.26
6	45.3	8.78	0.25
7	38.9	8.87	0.25
8	34.0	8.92	0.25
9	30.2	9.52	0.25
10	27.2	9.58	0.25
1G	---	24.36	0.20
MG	---	---	0.23

3.2 VERA Problem 4a-2D

Figure 7 shows the visualization of Problem 4a-2D, where the red pins denote 2.1% enriched fuel pins and green denotes 2.6%. It can also be seen that there are several pyrex rods (red annular regions with white center) in the 2.6% enriched assemblies.

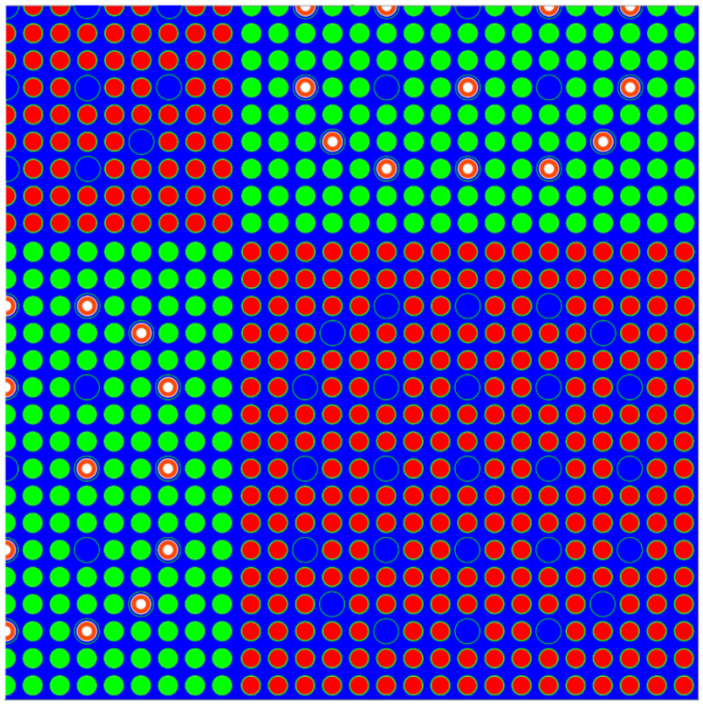


Figure 7. Geometry Visualization of VERA Problem 4a-2D [5]

Table 2 shows the results for Problem 4a-2D. In general, the trends are very similar to 2a with nearly a 2.8x speedup with 5 batches. This is not very surprising, particularly since both were executed in serial.

Table 2. Results for Problem 4a-2D

# Batch	Average Batch Size (pseudogroups)	Subgroup Time (sec)	Total Memory (GB)
1	272.0	80.22	1.99
2	136.0	78.25	1.43
3	90.7	77.55	1.25
4	68.0	79.52	1.16
5	54.4	80.59	1.11
6	45.3	77.62	1.07
7	38.9	84.96	1.04
8	34.0	79.43	1.04
9	30.2	81.97	1.04
10	27.2	84.11	1.04
1G	---	222.91	0.88
MG	---	---	1.02

3.3 VERA Problem 5a-2D

Figure 8 shows the P5-2D core layout, which consists of three different enrichment zones. Not shown in the figure are the baffle and reflector regions.

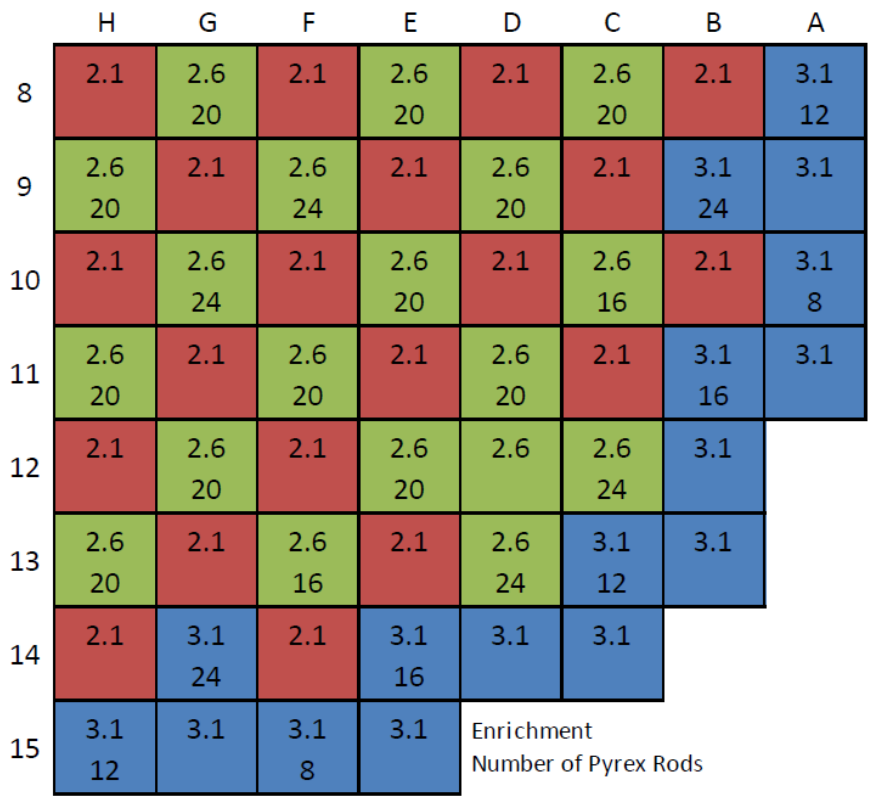


Figure 8. Geometry Visualization of VERA Problem 5a-2D [5]

Table 3 shows the results for 5a-2D, which were run on 73 processors on Titan [7]. The trends here are a little different and provide some interesting ideas. First of all, we see that 1 batch takes the most time. This is suspected to be an anomaly introduced by the spatial decomposition data passing, which now is passing buffers of data that contains a very substantial number of pseudogroups. This effect seems to be mitigated when increasing the batches, where an optimum point is reached around 5 batches. Because 5a-2D is a substantially larger problem than 2a and 4a-2D, the memory required is directly correlated, requiring nearly 68 GB with 1 batch, though spread across 73 processors. With 5 batches this is reduced considerably and is only marginally larger than the MG storage requirements by an average of 63 MB per process, indicating that the memory requirements for the improvements demonstrated in this report are indeed very small.

In terms of overall speedup, a nearly 3.9x speedup is observed. While a large portion of this is attributable to the lumped parameter approach, it is likely that this speedup is increased over what was observed in 2a and 4a-2D because the data passing is using larger buffers. In the 1G scheme, the angular flux data is passed using one-group buffers, whereas the new scheme passes using buffers of several groups.

Table 3. Results for Problem 5a-2D

# Batch	Average Batch Size (pseudogroups)	Subgroup Time (sec)	Total Memory (GB)
1	272.0	76.32	67.53
2	136.0	56.14	44.98
3	90.7	52.10	37.98
4	68.0	48.97	34.41
5	54.4	48.82	33.03
6	45.3	49.58	32.89
7	38.9	50.97	32.87
8	34.0	49.38	32.87
9	30.2	52.31	32.87
10	27.2	53.96	32.87
1G	---	188.95	23.74
MG	---	---	28.40

3.4 Reducing Azimuthal Angles

Table 4 shows the results from 5a-2D reducing the varying the number of azimuthal angles per octant with 16 used as the reference (and 16 were used for the eigenvalue sweep in all cases). As we can see, using only 4 azimuthal angles is too few, but using 8 angles yields a difference of only 14 pcm, with a negligible impact on pin powers. Additionally, running with 32 angles (for completeness) shows that there is nothing to gain by running with more. If 8 azimuthal angles were agreed to be acceptable, the overall speedup compared to the current approach would be slightly over 7x.

Table 4. Results for Problem 5a-2D Varying Azimuthal Angles for Shielding Calculation

Azimuthal Angles Per Octant	keff	Diff (pcm)	Pin Power		Time (sec)	Total Mem. (GB)
			RMS (%)	MAX (%)		
4	1.00233	-78	0.036	0.079	15.48	25.39
8	1.00297	-14	0.006	0.014	26.81	27.75
16	1.00311	0	---	---	48.82	33.03
32	1.00313	2	0.002	0.005	94.96	45.22

Appendix B contains the output from MPACTdiff.py [8] from VERA Problems 1 through 3 and 6 changing the number of azimuthal angles from 16 to 8 for the shielding calculation. Problem 1 demonstrates substantially more sensitivity with 30-40 pcm change observed, except for 1e, where a 0.005 ray spacing is used (all others use 0.05). Problems 2, 3, and 6 show similar sensitivity as 5a-2D with typically 10-20 pcm difference observed.

4. CONCLUSIONS

In this report, several different improvements were discussed and demonstrated, such as solving all pseudogroups concurrently, moving to a "lumped parameter" approach for MOC, and reducing the number of azimuthal angles per octant only for the shielding calculation. The performance

combining the first two improvements was assessed on three tests problems ranging from a single quarter assembly lattice to a quarter core 2D slice. Both serial cases demonstrated a speedup of around 2.8x, whereas the parallel 2D quarter core yielded nearly a 3.9x speedup. Additionally, the memory burden to achieve these gains was found to be tolerable, with a vast majority of the increase resulting from utilizing the multigroup kernel approach.

When considering reducing the number of azimuthal angles, it was found that using 8 azimuthal angles per octant compared to 16 which is typically used introduces only a 10-20 pcm difference, while providing additional speedup. This in combination with the previous two improvements yields over a 7x speedup on the 2D quarter core problem.

REFERENCES

1. S. Stimpson, B. Collins, and B. Kochunas, "MOC Efficiency Improvements Using a Jacobi Inscatter Approximation", Revision 1, CASL-I-2016-1056-001, CASL, March 30 (2016).
2. R. J. J. Stamm'ler et al., *HELIOS Methods*, Studsvik Scandpower (2003).
3. MPACT Team, "MPACT VERA Input User's Manual, Version 2.1.0," University of Michigan, Ann Arbor, MI, December 3 (2015).
4. K. S. Kim et al., "Development of a New 47-Group Library for the CASL Neutronics Simulators," *Proc. M&C 2015*, Nashville, Tennessee, April 19–23 (2015).
5. A. Godfrey, "VERA Core Physics Benchmark Progression Problem Specifications", Revision 4, CASL-U-2012-0131-004, CASL, August 29 (2014). http://www.casl.gov/docs/CASL-U-2012-0131-004.pdf
6. A. Yamamoto et al., "Derivation of Optimum Polar Angle Quadrature Set for the Method of Characteristics Based On Approximation Error for the Bickley Function," *Journal of Nuclear Science and Technology*, Vol. 4, No. 2, p. 129-136 (2007).
7. Oak Ridge Leadership Computing Facility. "Introducing Titan - The World's #1 Open Science Supercomputer" (2014), http://www.olcf.ornl.gov/titan/.
8. B. Collins, S. Stimpson, X. Wang, "Regression Suite Improvements in MPACT," Revision 0, CASL-U-2016-1053-000, CASL, December 8 (2015).

APPENDIX A. SAMPLE LUMPED PARAMETER DERIVATION

As a sample derivation for the lumped parameters A, B, and C, consider a simple pin cell problem typically consisting of 5 segments, which can be lumped into one, as in Figure A.1.

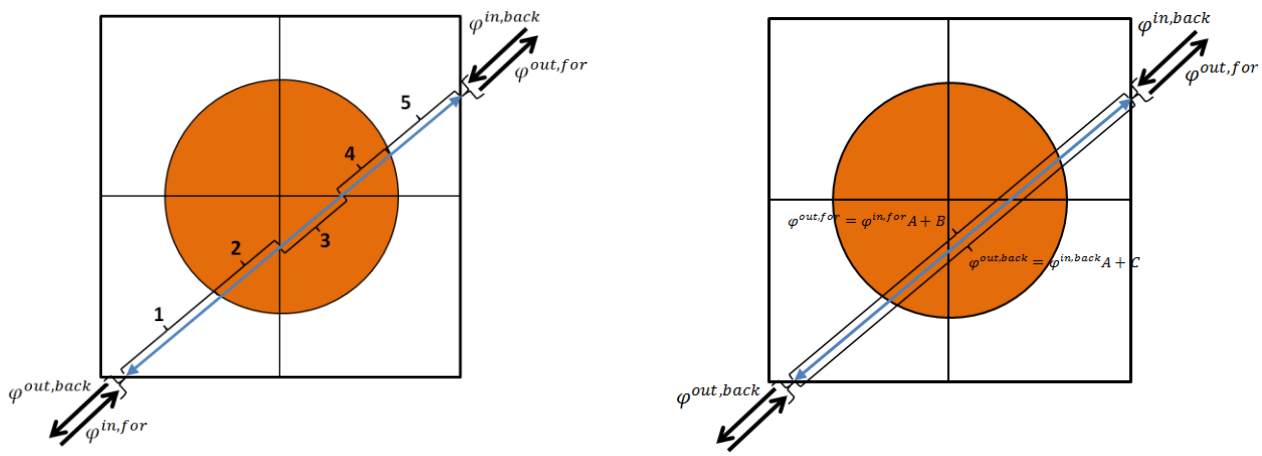


Figure A.1. Visualization of MOC Ray Tracing (left) and Lumped Parameter (right) on a Pin Cell

Below are the equations for the forward equations for each segment when sweeping.

<u>Forward Evaluation:</u>

1: $\varphi^1 = \varphi^{in,for} e^{-\Sigma_{t,1} l_1} + \frac{Q_1}{\Sigma_{t,1}}(1 - e^{-\Sigma_{t,1} l_1})$

2: $\varphi^2 = \varphi^1 e^{-\Sigma_{t,2} l_2} + \frac{Q_2}{\Sigma_{t,2}}(1 - e^{-\Sigma_{t,2} l_2})$

3: $\varphi^3 = \varphi^2 e^{-\Sigma_{t,3} l_3} + \frac{Q_3}{\Sigma_{t,3}}(1 - e^{-\Sigma_{t,3} l_3})$

4: $\varphi^4 = \varphi^3 e^{-\Sigma_{t,4} l_4} + \frac{Q_4}{\Sigma_{t,4}}(1 - e^{-\Sigma_{t,4} l_4})$

5: $\varphi^{out,for} = \varphi^4 e^{-\Sigma_{t,5} l_5} + \frac{Q_5}{\Sigma_{t,5}}(1 - e^{-\Sigma_{t,5} l_5})$

Substituting this into one another, an expression can be found directly between $\varphi^{in,for}$ and $\varphi^{out,for}$:

$$\varphi^{out,for} = \left(\left(\left(\left(\varphi^{in,for} e^{-\Sigma_{t,1} l_1} + \frac{Q_1}{\Sigma_{t,1}}(1 - e^{-\Sigma_{t,1} l_1})\right) e^{-\Sigma_{t,2} l_2} + \frac{Q_2}{\Sigma_{t,2}}(1 - e^{-\Sigma_{t,2} l_2})\right) e^{-\Sigma_{t,3} l_3} + \frac{Q_3}{\Sigma_{t,3}}(1 - e^{-\Sigma_{t,3} l_3})\right) e^{-\Sigma_{t,4} l_4} + \frac{Q_4}{\Sigma_{t,4}}(1 - e^{-\Sigma_{t,4} l_4})\right) e^{-\Sigma_{t,5} l_5} + \frac{Q_5}{\Sigma_{t,5}}(1 - e^{-\Sigma_{t,5} l_5})$$

Rearranging this, we can see that this relationship can be described by: $\varphi^{out,for} = \varphi^{in,for} A + B$

$$\varphi^{out,for} = \varphi^{in,for} e^{-\Sigma_{t,1} l_1} e^{-\Sigma_{t,2} l_2} e^{-\Sigma_{t,3} l_3} e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,5} l_5} + \left(\left(\left(\left(\frac{Q_1}{\Sigma_{t,1}} (1 - e^{-\Sigma_{t,1} l_1}) \right) e^{-\Sigma_{t,2} l_2} e^{-\Sigma_{t,3} l_3} e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,5} l_5} + \frac{Q_2}{\Sigma_{t,2}} (1 - e^{-\Sigma_{t,2} l_2}) e^{-\Sigma_{t,3} l_3} e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,5} l_5} \right) + \frac{Q_3}{\Sigma_{t,3}} (1 - e^{-\Sigma_{t,3} l_3}) e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,5} l_5} \right) + \frac{Q_4}{\Sigma_{t,4}} (1 - e^{-\Sigma_{t,4} l_4}) e^{-\Sigma_{t,5} l_5} \right) + \frac{Q_5}{\Sigma_{t,5}} (1 - e^{-\Sigma_{t,5} l_5})$$

where

$$A = e^{-\Sigma_{t,1} l_1} e^{-\Sigma_{t,2} l_2} e^{-\Sigma_{t,3} l_3} e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,5} l_5} \boxed{= \prod_{i=1}^{N_{seg}} e^{-\Sigma_{t,i} l_i}}$$

and

$$B = \left(\left(\left(\left(\frac{Q_1}{\Sigma_{t,1}} (1 - e^{-\Sigma_{t,1} l_1}) \right) e^{-\Sigma_{t,2} l_2} e^{-\Sigma_{t,3} l_3} e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,5} l_5} + \frac{Q_2}{\Sigma_{t,2}} (1 - e^{-\Sigma_{t,2} l_2}) e^{-\Sigma_{t,3} l_3} e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,5} l_5} \right) + \frac{Q_3}{\Sigma_{t,3}} (1 - e^{-\Sigma_{t,3} l_3}) e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,5} l_5} \right) + \frac{Q_4}{\Sigma_{t,4}} (1 - e^{-\Sigma_{t,4} l_4}) e^{-\Sigma_{t,5} l_5} \right) + \frac{Q_5}{\Sigma_{t,5}} (1 - e^{-\Sigma_{t,5} l_5}) \boxed{= \sum_{i=1}^{N_{seg}} \left(\frac{Q_i}{\Sigma_{t,i}} (1 - e^{-\Sigma_{t,i} l_i}) \prod_{j=i+1}^{N_{seg}} e^{-\Sigma_{t,j} l_j} \right)}$$

A similar procedure can be executed for the backward trace:

Backward Evaluation:

5: $\varphi^5 = \varphi^{in,back} e^{-\Sigma_{t,5} l_5} + \frac{Q_5}{\Sigma_{t,5}}(1 - e^{-\Sigma_{t,5} l_5})$

4: $\varphi^4 = \varphi^5 e^{-\Sigma_{t,4} l_4} + \frac{Q_4}{\Sigma_{t,4}}(1 - e^{-\Sigma_{t,4} l_4})$

3: $\varphi^3 = \varphi^4 e^{-\Sigma_{t,3} l_3} + \frac{Q_3}{\Sigma_{t,3}}(1 - e^{-\Sigma_{t,3} l_3})$

2: $\varphi^2 = \varphi^3 e^{-\Sigma_{t,2} l_2} + \frac{Q_2}{\Sigma_{t,2}}(1 - e^{-\Sigma_{t,2} l_2})$

1: $\varphi^{out,back} = \varphi^2 e^{-\Sigma_{t,1} l_1} + \frac{Q_1}{\Sigma_{t,1}}(1 - e^{-\Sigma_{t,1} l_1})$

$$\varphi^{out,back} = \left(\left(\left(\left(\varphi^{in,back} e^{-\Sigma_{t,5} l_5} + \frac{Q_5}{\Sigma_{t,5}}(1 - e^{-\Sigma_{t,5} l_5})\right) e^{-\Sigma_{t,4} l_4} + \frac{Q_4}{\Sigma_{t,4}}(1 - e^{-\Sigma_{t,4} l_4})\right) e^{-\Sigma_{t,3} l_3} + \frac{Q_3}{\Sigma_{t,3}}(1 - e^{-\Sigma_{t,3} l_3})\right) e^{-\Sigma_{t,2} l_2} + \frac{Q_2}{\Sigma_{t,2}}(1 - e^{-\Sigma_{t,2} l_2})\right) e^{-\Sigma_{t,1} l_1} + \frac{Q_1}{\Sigma_{t,1}}(1 - e^{-\Sigma_{t,1} l_1})$$

$$\varphi^{out,back} = \left(\left(\left(\left(\varphi^{in,back} e^{-\Sigma_{t,5} l_5} e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,3} l_3} e^{-\Sigma_{t,2} l_2} e^{-\Sigma_{t,1} l_1} + \frac{Q_5}{\Sigma_{t,5}}(1 - e^{-\Sigma_{t,5} l_5}) e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,3} l_3} e^{-\Sigma_{t,2} l_2} e^{-\Sigma_{t,1} l_1}\right) + \frac{Q_4}{\Sigma_{t,4}}(1 - e^{-\Sigma_{t,4} l_4}) e^{-\Sigma_{t,3} l_3} e^{-\Sigma_{t,2} l_2} e^{-\Sigma_{t,1} l_1}\right) + \frac{Q_3}{\Sigma_{t,3}}(1 - e^{-\Sigma_{t,3} l_3}) e^{-\Sigma_{t,2} l_2} e^{-\Sigma_{t,1} l_1}\right) + \frac{Q_2}{\Sigma_{t,2}}(1 - e^{-\Sigma_{t,2} l_2}) e^{-\Sigma_{t,1} l_1}\right) + \frac{Q_1}{\Sigma_{t,1}}(1 - e^{-\Sigma_{t,1} l_1})$$

The first term can be observed to be the same, but the second term will be different, overall conforming to $\varphi^{out,back} = \varphi^{in,back} A + C$:

$$A = e^{-\Sigma_{t,5} l_5} e^{-\Sigma_{t,4} l_4} e^{-\Sigma_{t,3} l_3} e^{-\Sigma_{t,2} l_2} e^{-\Sigma_{t,1} l_1} \boxed{= \prod_{i=1}^{N_{seg}} e^{-\Sigma_{t,i} l_i}}$$

$$C = \left(\left(\left(\left(\frac{Q_5}{\Sigma_{t,5}}(1-e^{-\Sigma_{t,5}l_5})e^{-\Sigma_{t,4}l_4}e^{-\Sigma_{t,3}l_3}e^{-\Sigma_{t,2}l_2}e^{-\Sigma_{t,1}l_1}\right)\right.\right.$$

$$\left.+\frac{Q_4}{\Sigma_{t,4}}(1-e^{-\Sigma_{t,4}l_4})e^{-\Sigma_{t,3}l_3}e^{-\Sigma_{t,2}l_2}e^{-\Sigma_{t,1}l_1}\right)+\frac{Q_3}{\Sigma_{t,3}}(1-e^{-\Sigma_{t,3}l_3})e^{-\Sigma_{t,2}l_2}e^{-\Sigma_{t,1}l_1}\Bigg)$$

$$\left.+\frac{Q_2}{\Sigma_{t,2}}(1-e^{-\Sigma_{t,2}l_2})e^{-\Sigma_{t,1}l_1}\right)+\frac{Q_1}{\Sigma_{t,1}}(1-e^{-\Sigma_{t,1}l_1})$$

$$\boxed{=\sum_{i=1}^{N_{seg}}\left(\frac{Q_i}{\Sigma_{t,i}}(1-e^{-\Sigma_{t,i}l_i})\prod_{j=1}^{i-1}e^{-\Sigma_{t,j}l_j}\right)}$$

In summary, we have the lumped parameters A, B, and C, which allow us to consolidate the segments into one operator:

$$\varphi^{out,for} = \varphi^{in,for}A + B$$
$$\varphi^{out,back} = \varphi^{in,back}A + C$$

where the parameters have the following equations:

$$A = \prod_{i=1}^{N_{seg}} e^{-\Sigma_{t,i}l_i}$$

$$B = \sum_{i=1}^{N_{seg}}\left(\frac{Q_i}{\Sigma_{t,i}}(1-e^{-\Sigma_{t,i}l_i})\prod_{j=i+1}^{N_{seg}} e^{-\Sigma_{t,j}l_j}\right)$$

$$C = \sum_{i=1}^{N_{seg}}\left(\frac{Q_i}{\Sigma_{t,i}}(1-e^{-\Sigma_{t,i}l_i})\prod_{j=1}^{i-1} e^{-\Sigma_{t,j}l_j}\right)$$

Alternatively, the B and C parameters can be given more directly once A is determined:

$$B = \varphi^{out,for} - \varphi^{in,for}A$$
$$C = \varphi^{out,back} - \varphi^{in,back}A$$

APPENDIX B. VERA PROBLEMS 1-3 RESULTS WITH 8 AZI. ANGLES PER OCTANT FOR SHIELDING

Case ID	State	Exposure (GWD/MT)	EFPD	dk (pcm)	dBoron (ppm)	dP RMS (%)	dP MAX (%)	dT RMS (C)	dT MAX (C)
1a	1	0.0000	---	37.74	---	0.000	0.000	---	---
1a_dep	1	0.0000	0.00	37.74	0.00	0.000	0.000	---	---
	2	0.3840	10.00	35.98	---	0.000	0.000	---	---
	3	0.7680	20.00	35.38	---	0.000	0.000	---	---
1b	1	0.0000	0.00	41.53	0.00	0.000	0.000	---	---
1c	1	0.0000	---	43.91	---	0.000	0.000	---	---
1d	1	0.0000	---	45.76	---	0.000	0.000	---	---
1e	1	0.0000	0.00	10.86	0.00	0.000	0.000	---	---
2a	1	0.0000	---	15.45	---	0.000	0.000	---	---
2a_dep	1	0.0000	0.00	15.45	0.00	0.000	0.000	---	---
	2	0.3841	10.00	14.79	---	0.000	0.000	---	---
	3	0.7681	20.00	14.52	---	0.000	0.010	---	---
	4	1.5362	40.00	14.06	---	0.000	0.010	---	---
	5	2.3044	60.00	13.65	---	0.010	0.020	---	---
2b	1	0.0000	---	19.70	---	0.000	0.000	---	---
2c	1	0.0000	---	20.74	---	0.000	0.000	---	---
2d	1	0.0000	---	21.57	---	0.000	0.000	---	---
2e	1	0.0000	---	14.72	---	0.000	0.000	---	---
2f	1	0.0000	---	13.95	---	0.000	0.000	---	---
2g	1	0.0000	---	10.59	---	0.000	0.010	---	---
2h	1	0.0000	---	10.28	---	0.000	0.000	---	---
2i	1	0.0000	---	15.57	---	0.000	0.000	---	---
2j	1	0.0000	---	13.96	---	0.000	0.000	---	---
2k	1	0.0000	---	14.28	---	0.000	0.000	---	---
2l	1	0.0000	0.00	12.77	0.00	0.000	0.010	---	---
2m	1	0.0000	0.00	11.33	0.00	0.000	0.010	---	---
2n	1	0.0000	0.00	11.61	0.00	0.000	0.000	---	---
2o	1	0.0000	---	13.67	---	0.000	0.000	---	---
2p	1	0.0000	---	11.87	---	0.000	0.000	---	---
2q	1	0.0000	0.00	20.14	0.00	0.000	0.000	---	---
3a	1	0.0000	0.00	16.17	0.00	0.000	0.010	---	---
3b	1	0.0000	0.00	14.75	0.00	0.000	0.010	---	---
p6	1	0.0000	0.00	18.92	0.00	0.050	0.070	0.190	0.30