ARTICLE TEMPLATE

One-cell inversions on modern GPUs for transient transport with higher order discretization

Joanna Piper Morgan[a], Ilham Variansyah[b] Damon McDougall[c] Todd S. Palmer[b], and Kyle E. Niemeyer[a]

[a]School of Mechanical Industrial and Manufacturing Engineering, Oregon State University, Corvallis, OR, USA; [b]School of Nuclear Science and Engineering, Oregon State University, Corvallis, OR, USA; [c]Advanced Micro Devices, Austin, TX, USA

ARTICLE HISTORY
Compiled September 16, 2024

ABSTRACT
blah blah blah

KEYWORDS
Sections; lists; figures; tables; mathematics; fonts; references; appendices

1. Introduction

To find deterministic solutions to the transient S_N (where N is the number of angles in a quadrature set) neutron transport equation typically requires iterative schemes to treat the scattering (and fission) source terms (Lewis and Miller 1984).

The source iteration (SI) method is commonly used to do this, often accompanied by preconditioners or synthetic accelerators, where the contribution to the solution from the scattering source is allowed to lag, while the angular flux is solved in every ordinate via transport sweeps through the spatial domain (Adams 1997). SI sweeps in Cartesian geometries are readily parallelized over the number of angles, as the source term is known from the previous iteration, allowing the angular flux in each ordinate to be computed independently. While any parallelization is a boon to performance, a scheme that is embarrassingly parallel over the dimension with the greatest number of degrees of freedom —space— may be advantageous. In a single spatial dimension SI is *annoying serial* in space and cannot be parallelized.

In higher spatial dimensions, many S_N production codes that implement SI use some kind of wavefront marching parallel algorithm also known as a Kockh-Baker-Alcouff scheme (Baker 2017). This is also called "full parallel sweeps" (FPS) in literature. In this scheme a sweep begins in a spatial location where all cell dependencies are known from boundary information (e.g. a corner). From there on a hypothetical 2D grid, the two nearest neighbor cells are now able to be computed independently, the next step would be 4 cells. This diagonally marching wavefront continues to march widening paralleling as many cells spatially as possible eventually saturating the number of work threads if the problem is large enough. On CPUs this has been shown to be

CONTACT Joanna Piper Morgan. Email: morgajoa@oregonstate.edu performant but this changing amount of work is not optimal on GPUs where KBA algorithms are also tricky to efficiently implement in domain decomposed where wave front propagation between boundaries can be tricky. While this work is concerned with 1 spatial dimension when analyzing the state of the art it is important to consider that this is done.

This has proven successful in modern transport applications on CPUs (e.g., PARTISN, which implements the Koch–Baker–Alcouffe or KBA algorithm).

The state of the art in deterministic S_N radiation transport is multi-group in energy distribution, diamond differencing first order space discretizations, backward euler time stepping, domain decomposition via parallel block jacobi, wave front marching schemes like KBA in higher spatial dimensions within a subdomain, and source iterations with diffusion synthetic acceleration and GMRES solvers. Some examples of production codes that implement this are Partisn, Ardra, Minerate, Capsaicin, Denovo, Silver Fir, etc.

When analyzing performance on GPUs the roofline performance model is often used. The roof is constructed by the communication or bandwidth (Bytes/S) and computation resources of a specific GPU for a specific numerical precision (floating operation points per second or FLOPS). As algorithms at their most optimized become performance limited by either bandwidth or compute resources. Thus when the performance of a given GPU device function is on the roofline it is the most optimized it could be.

Performance anylisys on GPUs of deterministic sweep codes is limited in the literature. That is espically the case when looking for roofline anylisys specifically. Roofline models have previously been used to evalueate some miniapps in the Monte Carlo world (Tramm and Siegel 2021; KWACK 2022).

One place there is data on is a miniapp called cripky (Kunen, Bailey, and Brown 2015). Kripke is the publicly distributed version of Ardra that has performance data available. Roofline anylisys of Kripke has been published for performance on an AMD MI200 GPUs (Wolfe and Lu 2022). Specific k We conject that this is due to the non-linear work load a wavefront marching schemes incurred. As the wave moves through the problem space in a given ordinant the number of cells to be solved in parallel can change dramatically. This kind of in kernel dynamic problem While our work is limited to a single spatial dimension thus avoiding anylisys of KBA algorithms this shows the state of the art in performance of deterministic code.

One Cell Inversions (OCI) (also called cell-wise parallel block Jacobi) is an alternative to SI where all angular fluxes in all ordinates within a cell are computed in a single linear algebra step. It assumes that the angular fluxes incident on the surfaces of the cell are known from a previous iteration. OCI allows for parallelizing over the number of cells as each cell is solved independently of the others in a parallel block Jacobi scheme. Block

Rosa, Warsa, and Perks (2013) previously investigated OCI as a potentially superior iterative scheme over SI on vectorized architectures. They supposed that because of the parallelism over the dominant domain, inherit data locality, ability to take advantage of LAPACK type libraries, highly floppy operations present in an OCI type algorithm, it might out-perform a SI based implementation in wall clock runtime. The study was conducted on state of the art at the time RoadRunner super computer and took advantage of its 64 bit PowerXCell vectorized accelerator based. While not specifically designed as a GPU, notably versions of the Cell Broadband Engine architecture would be used on the Sony Playstation 3 gaming console. Rosa et. al. implemented OCI in a 2D, multi-group, steady state, code using first order diamond differencing to discretize space, a multi-group scheme in energy distribution, and parallel block Jacobi and

Gauss-Sidle iterations. The difference between parallel block Jacobi and gauss sidel is blah. For the sake of brevity we consider both to be versions of OCI as the principle is the same.

The authors concluded that the acceleration seen per iteration in an OCI implementation was not enough to make up fo the decay in convergence rate that OCI incurs. Because there is no communication of information between cells within an iteration, OCI can require more iterations to converge to a solution for some of problems. Specifically, as cellular optical thickness goes down, OCI's relative performance degrades. Figure 1 illustrates this behavior, showing the spectral radii of the two iteration schemes as a function of cell thickness (in mean free path) and the scattering ratio. These values were computed numerically from an infinite medium problem (via reflecting boundaries) using steady-state calculations in S_4. Gauss Legendre quadrature was used in all presented explorations. The smaller the spectral radius, the faster a method is converging. The spectral radius for SI depends strongly on the scattering ratio, and for problems that are many mean free paths in size, it is nearly independent of cell optical thickness. The spectral radius of OCI decreases substantially as the optical thickness of the cells increases.

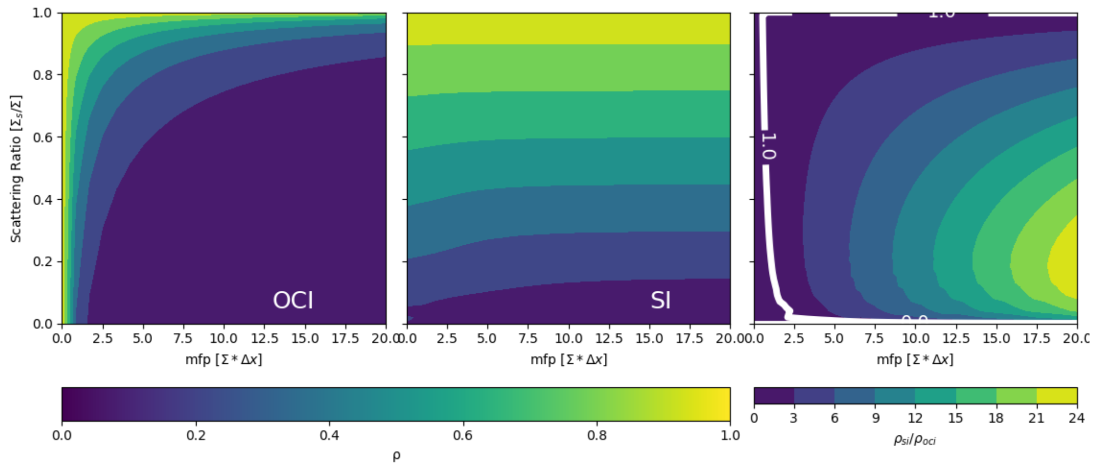

Figure 1. Spectral radii (**ρ**) of OCI (left) and SI (middle) and the ratio between the two (right), where **Σ** is the total cross section, **Δx** is the cell width, and **Σ_s** is the scattering cross section

Rosa et. al. concluded by suggesting that future developments in GPGPU accelerators might overcome this hurtle.

Other investigations have explored OCI as an acceleration scheme for SI (Anistratov and Azmy 2015; Hoagland and Azmy 2021) and a solution to the Integral transport matrix method (Schunert et al. 2018). Previous investigations of OCI have been limited to single order discretization schemes and steady state computations.

When solving discrete ordnance problems for transient effects many codes have implemented a Crank-Nicholson or backward Euler time stepping. These schemes are non-intrusive often looking like an additional time marching loop around the already implemented transport infrastructure. Regardless of the time stepping method an OCI iterative scheme might come with some added befits when used in a transient anylisys. Returning to Fig. 1, since both dimensions are governed by relationships with the total cross section (Σ), altering that value will impact convergence behavior. As the scattering ratio decreases, both iterative schemes converge more quickly. However, the spectral radius of OCI also decreases with increasing optical thickness, which is an added benefit. When solving optically thick and highly scattering problems, small increases in Σ may drastically improve the relative performance of OCI in comparison with SI. Physically this can be understood as single particles living in single cells for many more iterations Analytically this is shown in the derivations in section 2.

Time step and cellular optical thickness are inversely proportional to each other, meaning a smaller time step will yield a larger effective total cross section, thus theoretically improving the spectral radius. This behavior is not expected to happen in source iterations as SI does not directly depend on cellular optical thickness—behaving linearly in that dimension. This coupled with the vast improvements in GPGPU accelerators in the past decade since Rosa, Warsa, and Perks (2013) conducted their investigations, along with the ability to explore higher order spatial and transient discretization schemes motivates our work.

We have previously shown for mono-energetic problems that when space is the dominant dimension (e.g. many more spatial cells then angles in quadrature) a GPGPU implementation of OCI will out perform a similarly implemented version of SI in wall clock runtime (Morgan et al. 2023) in all but the highest scattering problems. In this work we extend analysis for energy dependent problems using optimized GPU compute kernels and conduct in-depth anylisys of how time dependence alters convergence rate of OCI and SI. We show derivations and Fourier anylisys of our time and space discretization schemes showing we produce a second order scheme in both dimensions for both OCI and SI. We use analytic benchmark problems, method of manufactured solutions, and multi-group Monte Carlo simulations to verify the solutions provided by either iteration scheme and confirm our discretization scheme is converging to the correct solution. We implement both OCI and SI iterations using this discretization scheme with state of the art vendor supplied LAPACK solvers in `C++`. Finally we explore transient convergence behavior of OCI as compared to SI.

2. Derivation and Anylisys of Higher Order Discretization Scheme

To further improve the GPU parallel performance, we investigate higher-order discretization methods, particularly the robust, second-order accurate spatial discretization method simple corner balance (SCB) (Adams 1997) and the (also) robust, second-order accurate time discretization method multiple balance (MB) (Variansyah, Larsen, and Martin 2021). In coupling this higher temporal accuracy scheme with an iterative method that can be efficiently solved, we hope to optimize the ratio of compute work to communication work to better suit the numerical method for GPUs.

We start by deriving the discretized slab geometry transport equations—SCB in space, MB in time, S_N in angle—show that the resulting method MB-SCB is unconditionally stable via Fourier analysis, and then derive the respective iterative systems.

2.1. *Derivation of MB-SCB*

We begin with the time-dependent, isotropic scattering slab geometry S$_N$ transport equations with an isotropic source:

$$\begin{aligned}
\frac{1}{v_g}\frac{\partial \psi_{m,g}(x,t)}{\partial t} + \mu_m \frac{\partial \psi_{m,g}(x,t)}{\partial x} + \Sigma_g(x)\psi_{m,g}(x,t) \\
= \frac{1}{2}\left(\sum_{g'=0}^{G} \Sigma_{s,g'\to g}(x) \sum_{n=1}^{N} w_n \psi_{n,g'}(x,t) + Q_g(x,t)\right) \ , \\
g = 1 \dots G \ , \qquad m = 1 \dots N \ , \qquad t > 0 \ , \qquad x \in [0, X] \ , \quad (1)
\end{aligned}$$

where ψ is the angular flux, t is time, x is location, v is velocity, w_m is angular quadrature weight, μ_m is the angular quadrature ordinate, m is the quadrature index, and Q is the isotropic material source. The initial and boundary conditions are prescribed angular flux distributions:

$$\psi_{m,g}(x,0) = \psi_{m,0}(x), \qquad m = 1 \dots N \ ,$$

$$\psi_{m,g}(0,t) = \psi_{m,L}(t), \qquad \mu_m > 0 \ ,$$

$$\psi_{m,g}(X,t) = \psi_{m,R}(t), \qquad \mu_m < 0 \ .$$

We discretize these equations in time using the MB approach (Variansyah, Larsen, and Martin 2021):

$$\begin{aligned}
\frac{1}{v_g}\left(\frac{\psi_{m,g,k+1/2}(x) - \psi_{m,g,k-1/2}(x)}{\Delta t}\right) + \mu_m \frac{\partial \psi_{m,g,k}(x)}{\partial x} + \Sigma_g(x)\psi_{m,g,k}(x) \\
= \frac{1}{2}\left(\sum_{g'=0}^{G} \Sigma_{s,g'\to g}(x) \sum_{n=1}^{N} w_n \psi_{n,g,k}(x) + Q_{g,k}(x)\right) \ , \quad (2a)
\end{aligned}$$

$$\begin{aligned}
\frac{1}{v_g}\frac{\psi_{m,g,k+1/2}(x) - \psi_{m,g,k}(x)}{\Delta t/2} + \mu_m \frac{\partial \psi_{m,g,k+1/2}(x)}{\partial x} + \Sigma_g(x)\psi_{m,k+1/2}(x) \\
= \frac{1}{2}\left(\sum_{g'=0}^{G} \Sigma_{s,g'\to g}(x) \sum_{n=1}^{N} w_n \psi_{n,k+1/2}(x) + Q_{g,k+1/2}(x)\right) \ , \quad (2b)
\end{aligned}$$

where Δt is the time step size, k indexes time-average quantities, and $k \pm 1/2$ indexes time-edge quantities. Then, we discretize in space using SCB, which involves a spatial integration over the right and left halves of a spatial cell:

$$\frac{\Delta x_j}{2}\frac{1}{v_g}\left(\frac{\psi_{m,g,k+1/2,j,L}-\psi_{m,g,k-1/2,j,L}}{\Delta t}\right) + \mu_m\left[\frac{(\psi_{m,g,k,j,L}+\psi_{m,g,k,j,R})}{2}-\psi_{m,g,k,j-1/2}\right] + \frac{\Delta x_j}{2}\Sigma_j\psi_{m,g,k,j,L} = \frac{\Delta x_j}{2}\frac{1}{2}\left(\sum_{g'=0}^{G}\Sigma_{s,j,g'\to g}(x)\sum_{n=1}^{N}w_n\psi_{n,g,k,j,L}+Q_{k,j,L}\right) , \tag{3a}$$

$$\frac{\Delta x_j}{2}\frac{1}{v}\left(\frac{\psi_{m,k+1/2,j,R}-\psi_{m,k-1/2,j,R}}{\Delta t}\right) + \mu_m\left[\psi_{m,k,j+1/2}-\frac{(\psi_{m,k,j,L}+\psi_{m,k,j,R})}{2}\right] + \frac{\Delta x_j}{2}\Sigma_j\psi_{m,k,j,R} = \frac{\Delta x_j}{2}\frac{1}{2}\left(\sum_{g'=0}^{G}\Sigma_{s,j,g'\to g}\sum_{n=1}^{N}w_n\psi_{n,g',k,j,R}+Q_{k,j,R}\right) , \tag{3b}$$

$$\frac{\Delta x_j}{2}\frac{1}{v_g}\left(\frac{\psi_{m,g,k+1/2,j,L}-\psi_{m,g,k,j,L}}{\Delta t/2}\right) + \mu_m\left[\frac{(\psi_{m,g,k+1/2,j,L}+\psi_{m,g,k+1/2,j,R})}{2}-\psi_{m,g,k+1/2,j-1/2}\right] + \frac{\Delta x_j}{2}\Sigma_j\psi_{m,g,k+1/2,j,L} = \frac{\Delta x_j}{2}\frac{1}{2}\left(\sum_{g'=0}^{G}\Sigma_{s,j,g'\to g}\sum_{n=1}^{N}w_n\psi_{n,g',k+1/2,j,L}+Q_{k+1/2,j,L}\right) , \tag{3c}$$

$$\frac{\Delta x_j}{2}\frac{1}{v_g}\left(\frac{\psi_{m,g,k+1/2,j,R}-\psi_{m,g,k,j,R}}{\Delta t/2}\right) + \mu_m\left[\psi_{m,g,k+1/2,j+1/2}-\frac{(\psi_{m,g,k+1/2,j,L}+\psi_{m,g,k+1/2,j,R})}{2}\right] + \frac{\Delta x_j}{2}\Sigma_j\psi_{m,g,k+1/2,j,R} = \frac{\Delta x_j}{2}\frac{1}{2}\left(\sum_{g'=0}^{G}\Sigma_{s,j,g'\to g}\sum_{n=1}^{N}w_n\psi_{n,g',k+1/2,j,R}+Q_{k+1/2,j,g,R}\right) , \tag{3d}$$

where Δx is the cell width, j is the spatial index, R is the right hand side sub cell division, and L is the left. These equations contain the first of the two simple spatial closures—the angular flux at the cell midpoint is a simple average of the two half-cell average quantities:

$$\psi_{m,g,k}(x_j) = \frac{(\psi_{m,k,j,L} + \psi_{m,k,j,R})}{2} , \tag{4a}$$

$$\psi_{m,g,k+1/2}(x_j) = \frac{(\psi_{m,k+1/2,j,L} + \psi_{m,k+1/2,j,R})}{2} . \tag{4b}$$

The second is an *upstream* prescription for the cell-edge angular flux:

$$\psi_{m,g,k,j+1/2} = \begin{cases} \psi_{m,g,k,j,R}, & \mu_m > 0, \\ \psi_{m,g,k,j+1,L}, & \mu_m < 0 , \end{cases} \tag{5a}$$

$$\psi_{m,g,k+1/2,j+1/2} = \begin{cases} \psi_{m,g,k+1/2,j,R}, & \mu_m > 0, \\ \psi_{m,g,k+1/2,j+1,L}, & \mu_m < 0 . \end{cases} \tag{5b}$$

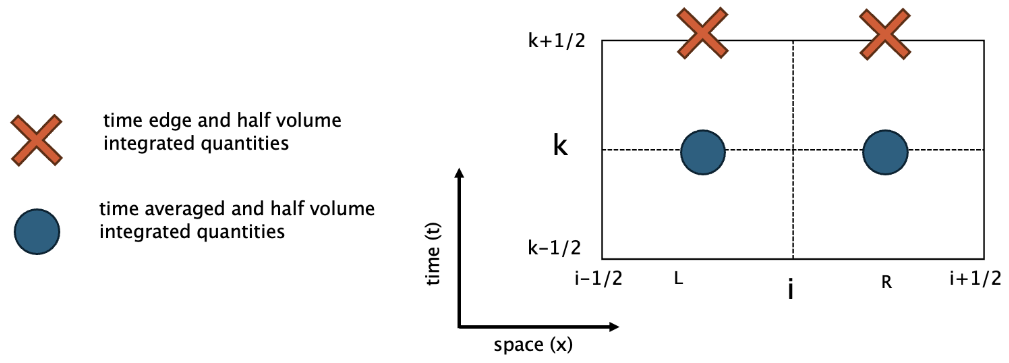
time edge and half volume integrated quantities
time averaged and half volume integrated quantities
k+1/2
k
k-1/2
i-1/2
L
i
R
i+1/2
time (t)
space (x)

Figure 2. The discretization stencil

Figure 2 shows the stencil location for angular flux and source terms.

2.2. *Fourier Analysis of Numerical Method*

To ensure that our combination of higher-order discretization schemes is still unconditionally stable, we performed a Fourier analysis to find the slowest-converging mode of the method. Assuming no scattering, no sources, mono-energetic problem, and make physical assumptions (only constant positive values of Δx, Δt, Σ, and v) we derived SCB-MB's eigenfunction and numerically solve it. We start by defining our Fourier ansätze as,

$$\psi_{k+1/2,j,L} = \theta^{k+1} a e^{i\omega j} \qquad \psi_{k+1/2,j,R} = \theta^{k+1} b e^{i\omega j} \tag{6a}$$

$$\psi_{k,j,L} = \theta^{k} c e^{i\omega j} \qquad \psi_{k,j,R} = \theta^{k} d e^{i\omega j} \tag{6b}$$

$$\psi_{k-1/2,j,L} = \theta^k a e^{i\omega j} \qquad\qquad \psi_{k-1/2,j,R} = \theta^k b e^{i\omega j} \tag{6c}$$

where k is the mode number, i is the imaginary number, ω is the frequency of the mode, θ is the , and j is the specific cell. Substituting our ansätze into Eqs. (3a), (3b), (3c), and (3d), respectively, we simplify to form:

$$\frac{\Delta}{x}\frac{1}{v\Delta t}(\theta a - a) + \mu\left(\frac{c+d}{2} - de^{-i\omega}\right) + \frac{\Sigma\Delta x}{2}c = 0 \tag{7a}$$

$$\frac{\Delta}{x}\frac{1}{v\Delta t}(\theta b - b) + \mu\left(ce^{i\omega} - \frac{c+d}{2}\right) + \frac{\Sigma\Delta x}{2}d = 0 \tag{7b}$$

$$\frac{\Delta}{x}\frac{1}{v\Delta t}(\theta a - c) + \theta\mu\left(\frac{a+b}{2} - be^{-i\omega}\right) + \frac{\Sigma\Delta x}{2}a\theta = 0 \tag{7c}$$

$$\frac{\Delta}{x}\frac{1}{v\Delta t}(\theta b - d) + \mu\left(de^{i\omega} + \frac{c+d}{2}\right) + \frac{\Sigma\Delta x}{2}a\theta = 0\ , \tag{7d}$$

combining Eq. (7a) into (7b):

$$\begin{bmatrix} c \\ d \end{bmatrix} = \boldsymbol{A}^{-1}\frac{\Delta x}{2}\frac{1}{v\Delta t}(1-\theta)\begin{bmatrix} a \\ b \end{bmatrix}\ , \tag{8}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \frac{\mu}{2} + \frac{\Sigma\Delta x}{2} & \mu(\frac{1}{2} - e^{-i\omega}) \\ \mu(e^{i\omega} - \frac{1}{2}) & -\frac{\mu}{2} + \frac{\Sigma\Delta x}{2} \end{bmatrix}\ .$$

Then, doing the same with Eq. (7c) into (7d):

$$\boldsymbol{B}\theta\begin{bmatrix} a \\ b \end{bmatrix} = \frac{\Delta x}{2}\frac{2}{v\Delta t}\left(\boldsymbol{A}^{-1}\frac{\Delta x}{2}\frac{1}{v\Delta t}(1-\theta)\right)\ , \tag{9}$$

where

$$\boldsymbol{B} = \begin{bmatrix} \frac{\Delta x}{2}\frac{2}{c\Delta t} + \frac{\mu}{2} + \frac{\Sigma\Delta x}{2} & \mu(\frac{1}{2} - e^{-i\omega}) \\ \mu(e^{i\omega} - \frac{1}{2}) & \frac{\Delta x}{2}\frac{2}{c\Delta t} + \frac{\mu}{2} + \frac{\Sigma\Delta x}{2} \end{bmatrix}\ .$$

Substituting Eq. (9) into (8) and rearranging:

$$(\boldsymbol{B} + \gamma\boldsymbol{A}^{-1})\theta\begin{bmatrix} a \\ b \end{bmatrix} = \frac{\Delta x}{2}\frac{2}{v\Delta t}\boldsymbol{A}^{-1}\frac{\Delta x}{2}\frac{1}{v\Delta t} \tag{10}$$

where $\gamma = \frac{\Delta x}{v\Delta t}\frac{\Delta x}{2v\Delta t}$ This can then be more appropriately posed as an eigenfunction:

$$\theta\begin{bmatrix} a \\ b \end{bmatrix} = (\boldsymbol{B} + \gamma\boldsymbol{A}^{-1})(\gamma\boldsymbol{A}^{-1})\begin{bmatrix} a \\ b \end{bmatrix} = \Lambda\begin{bmatrix} a \\ b \end{bmatrix}\ . \tag{11}$$

When this system is solved for its eigenvalues (Λ) numerically it shows that a MB-SCB scheme is unconditionally stable over $\mu \in [-1, 1]$.

We found that the MB-SCB scheme is unconditionally stable over $\mu \in [-1, 1]$. While experimenting with this method we did find that, under some conditions, it can produce negative fluxes; however, the negative flux oscillations were critically damped and dissipated with time.

2.3. *Source Iteration (SI)*

In the traditional SI, the scattering source is presumed known from a previous iteration, which leads to the following set of equations to be solved in transport "sweeps". This means that new estimates of both the end of time-step value of angular flux and time-averaged angular flux are computed together in each cell.

For SCB in slab geometry, this means there is a local 4×4 matrix to be solved in each cell. For $\mu_m > 0$, this equation has the form:

$$\mathbf{A}^{+} \begin{bmatrix} \psi_{m,g,k,j,L} \\ \psi_{m,g,k,j,R} \\ \psi_{m,g,k+1/2,j,L} \\ \psi_{m,g,k+1/2,j,R} \end{bmatrix} = \mathbf{b}^{+} \ . \tag{12}$$

Here, the $\mathbf{A}^{+}$ matrix has the following structure and element definitions:

$$\mathbf{A_{j,n,g}}^{+} = \begin{bmatrix} \frac{\mu_m + \Delta x_j \Sigma_{j,g}}{2} & \frac{\mu_m}{2} & \frac{\Delta x_j}{2 v_g \Delta t} & 0 \\ -\frac{\mu_m}{2} & \frac{\mu_m + \Delta x_j \Sigma_{j,g}}{2} & 0 & \frac{\Delta x_j}{2 v_g \Delta t} \\ -\frac{\Delta x_j}{v_g \Delta t} & 0 & \frac{\Delta x_j}{v_g \Delta t} + \frac{\mu_m + \Delta x_j \Sigma_{j,g}}{2} & \frac{\mu_m}{2} \\ 0 & -\frac{\Delta x_j}{v_g \Delta t} & -\frac{\mu_m}{2} & \frac{\Delta x_j}{v_g \Delta t} + \frac{\mu_m + \Delta x_j \Sigma_{j,g}}{2} \end{bmatrix} \tag{13}$$

and $\mathbf{b}^{+}$ is given by

$$\mathbf{b}^{+} = \begin{bmatrix} \frac{\Delta x_j}{4} \left(\sum_{g'=0}^{G} \Sigma_{s,g' \to g,j} \phi^{(l)}_{k,g,j,L} + Q_{k,g,j,L} \right) + \frac{\Delta x_j}{2 v \Delta t} \psi_{m,k-1/2,j,L} + \mu_m \psi_{m,k,j-1,R} \\ \frac{\Delta x_j}{4} \left(\sum_{g'=0}^{G} \Sigma_{s,g' \to g,j} \phi^{(l)}_{k,g,j,R} + Q_{k,g,j,R} \right) + \frac{\Delta x_j}{2 v \Delta t} \psi_{m,k-1/2,g,j,R} \\ \frac{\Delta x_j}{4} \left(\sum_{g'=0}^{G} \Sigma_{s,g' \to g,j} \phi^{(l)}_{k+1/2,g,j,L} + Q_{k+1/2,g,j,L} \right) + \mu_m \psi_{m,k+1/2,g,j-1,R} \\ \frac{\Delta x_j}{4} \left(\sum_{g'=0}^{G} \Sigma_{s,g' \to g,j} \phi^{(l)}_{k+1/2,g,j,R} + Q_{k+1/2,g,j,R} \right) \end{bmatrix} . \tag{14}$$

Similarly for $\mu_m < 0$, this equation has the form:

$$\mathbf{A}^{-} \begin{bmatrix} \psi_{m,g,k,j,L} \\ \psi_{m,g,k,j,R} \\ \psi_{m,g,k+1/2,j,L} \\ \psi_{m,g,k+1/2,j,R} \end{bmatrix} = \mathbf{b}^{-} \ . \tag{15}$$

Here, the $\mathbf{A}^-$ matrix has the following structure and element definitions:

$$\mathbf{A}^- = \begin{bmatrix} \frac{-\mu_m+\Delta x_j \Sigma_j}{2} & \frac{\mu_m}{2} & \frac{\Delta x_j}{2 v_g \Delta t} & 0 \\ -\frac{\mu_m}{2} & \frac{-\mu_m+\Delta x_j \Sigma_{g,j}}{2} & 0 & \frac{\Delta x_j}{2 v \Delta t} \\ -\frac{\Delta x_j}{v_g \Delta t} & 0 & \frac{\Delta x_j}{v_g \Delta t} + \frac{-\mu_m+\Delta x_j \Sigma_{g,j}}{2} & \frac{\mu_m}{2} \\ 0 & -\frac{\Delta x_j}{v \Delta t} & -\frac{\mu_m}{2} & \frac{\Delta x_j}{v_g \Delta t} + \frac{-\mu_m+\Delta x_j \Sigma_{g,j}}{2} \end{bmatrix} \tag{16}$$

and $\mathbf{b}^-$ is given by

$$\mathbf{b}^- = \begin{bmatrix} \frac{\Delta x_j}{4}\left(\sum_{g'=0}^{G} \Sigma_{s,g'\to g,j} \phi^{(l)}_{k,g,j,L} + Q_{g,k,j,L}\right) + \frac{\Delta x_j}{2 v_g \Delta t}\psi_{m,g,k-1/2,j,L} \\ \frac{\Delta x_j}{4}\left(\sum_{g'=0}^{G} \Sigma_{s,g'\to g,j} \phi^{(l)}_{g,k,j,R} + Q_{g,k,j,R}\right) + \frac{\Delta x_j}{2 v_g \Delta t}\psi_{m,k-1/2,j,R} - \mu_m \psi_{m,k,j+1,L} \\ \frac{\Delta x_j}{4}\left(\sum_{g'=0}^{G} \Sigma_{s,g'\to g,j} \phi^{(l)}_{g,k+1/2,j,L} + Q_{g,k+1/2,j,L}\right) \\ \frac{\Delta x_j}{4}\left(\sum_{g'=0}^{G} \Sigma_{s,g'\to g,j} \phi^{(l)}_{g,k+1/2,j,R} + Q_{g,k+1/2,j,R}\right) - \mu_m \psi_{m,g,k+1/2,j+1,L} \end{bmatrix} . \tag{17}$$

After sweeping the mesh cells in the appropriate directions for each angle in the quadrature set, the scalar flux vector can be updated via

$$\begin{bmatrix} \phi_{g,k,j,L} \\ \phi_{g,k,j,R} \\ \phi_{g,k+1/2,j,L} \\ \phi_{g,k+1/2,j,R} \end{bmatrix} = \sum_{n=1}^{N} w_n \begin{bmatrix} \psi_{n,g,k,j,L} \\ \psi_{n,g,k,j,R} \\ \psi_{n,g,k+1/2,j,L} \\ \psi_{n,g,k+1/2,j,R} \end{bmatrix} , \tag{18}$$

and the source iteration can continue until this scalar flux vector ceases changing between iterations. After converging, the simulation can move to the next time step.

2.4. *One-Cell Inversion (OCI)*

In OCI, the scattering source is subtracted to the left-hand side, and the information that comes from cells other than cell j is assumed to be known from a previous iteration. This means that all $4N$ angular fluxes (N angles, L and R, k and $k+1/2$) are computed simultaneously in cell j. For SCB in slab geometry, this means there is a local $4N \times 4N$ matrix to be solved in each cell.

$$(\mathbf{A} - \mathbf{S})\,\mathbf{\Psi} = \mathbf{c} \ , \tag{19}$$

where,

$$\mathbf{A_g} = \begin{bmatrix} \mathbf{A}_{1,g} & 0 & \cdots & 0 \\ 0 & \mathbf{A}_{2,g} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{A}_{N,g} \end{bmatrix} . \tag{20}$$

Here, the $\mathbf{A}_{m,g}$ matrix has the following structure and element definitions:

$$\mathbf{A}_{m,g} = \begin{cases} \mathbf{A}_g^+ & \mu_m > 0 \\ \mathbf{A}_g^- & \mu_m < 0 \end{cases} . \tag{21}$$

The scattering source $\mathbf{S_{g'\to g}}$ is defined by

$$S_{l,n} = \frac{\Delta x_j \Sigma_{s,g'\to g,j}}{4} w_n \ , \tag{22}$$

$\boldsymbol{\Psi}$ is given by:

$$\boldsymbol{\Psi}_{\mathbf{g}} = \left[\bar{}_{1,g} \ \bar{}_{2,g} \ \cdots \ \bar{}_{N,g}\right]^T \ . \tag{23}$$

where

$$\bar{}_n = \left[\psi_{n,k,j,L} \ \psi_{n,k,j,R} \ \psi_{n,k+1/2,j,L} \ \psi_{n,k+1/2,j,R}\right]^T \ , \tag{24}$$

and $\mathbf{c}$ is given by

$$\mathbf{c} = \left[\mathbf{c}_1 \ \mathbf{c}_2 \ \cdots \ \mathbf{c}_N\right]^T \ , \tag{25}$$

where

$$\mathbf{c}_m = \begin{cases} \mathbf{c}^+ & \mu_m > 0 \\ \mathbf{c}^- & \mu_m < 0 \end{cases} , \tag{26}$$

$$\mathbf{c}^+ = \begin{bmatrix} \frac{\Delta x_j}{4} Q_{k,j,L} + \frac{\Delta x_j}{2v\Delta t}\psi_{m,k-1/2,j,L} + \mu_m \psi^{(l)}_{m,k,j-1,R} \\ \frac{\Delta x_j}{4} Q_{k,j,R} + \frac{\Delta x_j}{2v\Delta t}\psi_{m,k-1/2,j,R} \\ \frac{\Delta x_j}{4} Q_{k+1/2,j,L} + \mu_m \psi^{(l)}_{m,k+1/2,j-1,R} \\ \frac{\Delta x_j}{4} Q_{k+1/2,j,R} \end{bmatrix} , \tag{27}$$

$$\mathbf{c}^- = \begin{bmatrix} \frac{\Delta x_j}{4} Q_{k,j,L} + \frac{\Delta x_j}{2v\Delta t}\psi_{m,k-1/2,j,L} \\ \frac{\Delta x_j}{4} Q_{k,j,R} + \frac{\Delta x_j}{2v\Delta t}\psi_{m,k-1/2,j,R} - \mu_m \psi^{(l)}_{m,k,j+1,L} \\ \frac{\Delta x_j}{4} Q_{k+1/2,j,L} \\ \frac{\Delta x_j}{4} Q_{k+1/2,j,R} - \mu_m \psi^{(l)}_{m,k+1/2,j+1,L} \end{bmatrix} . \tag{28}$$

Where l indexes solutions from the previous iteration. One cell inversion iterations continue until this angular flux vector ceases changing between iterations. After convergence, the time-step counter is incremented and the within time-step process can be repeated.

The structure of the within cell system of equations that is formed is shown for a hypothetical 2 group 4 angle problem in figure (blah)

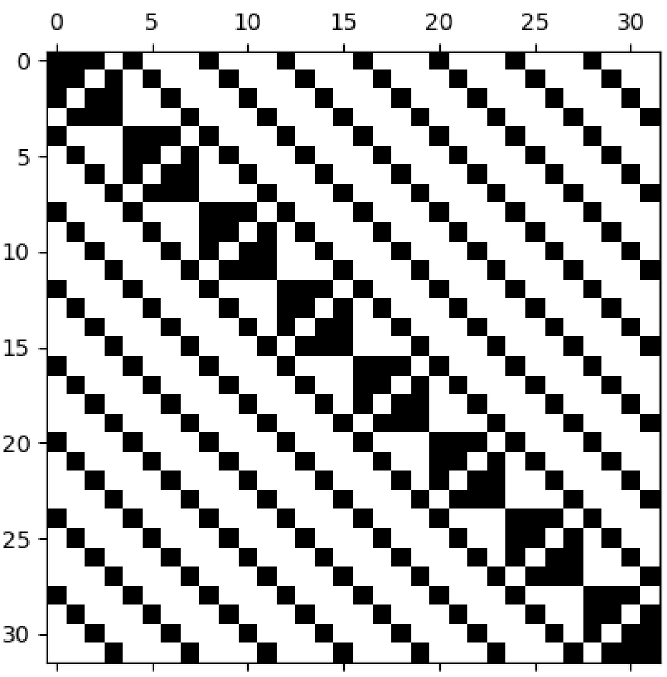

Figure 3. blah

2.5. *Verification using Method of Manufactured Solutions*

To verify both our discretization and iterative schemes are converging to the correct solution we first ran canonical analytic (source free pure absorber and infinite homogenous medium problems). When these produced the correct result we moved on to more complex verification tools to ensure correctness—the Method of manufactured solutions and an over resolved Monte Carlo solution. Finding benchmark problems that simulate transient, energy dependent, single dimension solutions is difficult. So we used the Method of Manufactured solutions to derive our own benchmark (Roy 2005). It has previously been used to verify other deterministic neutron transport codes including the adjoint solver MOOSE (Wang, Martin, and Collins 2018) and the blah solver blah (Gleicher II et al. 2012).

The method of manufactured solutions follow a backward solving technique where a solution is made-up and then inserted into the governing equation and solved for a source term. This solution is usually a continuous function, can be non-physical, and advantageously defined to ease computation. Then the now solved for source can be inserted into the solver We insert this solution to the NTE and solve for the source term. This can be done analytically tho we chose to use a Computer Algebra System (CAS) provide by SymPY (Meurer et al. 2017) to derive these solutions Now we use the . With our space and time discretization things can get tricky as some terms are cell averaged and some are cell edge. When this is carefully done we can derive our source and reinsert into the source term of our solver. We then let our solver itterate to convergence and compare the manufactured solution and our solver soultion.

To derive our method of manufactured solution we start with a conditions functional representation of the angular flux in angle, energy, space and time

$$\psi_1(x,t,\mu) = (1-\mu^2)\sin(\pi x)e^{-t} \tag{29a}$$
$$\psi_2(x,t,\mu) = (1-\mu^2)(-x^4+x+1)e^{-t/2} \tag{29b}$$

The range we will be solving these equations is $x \in [0,1]$, $t \in [0,5]$ and $\mu \in [-1,1]$. Now we take 29a and 29b and insert them into the S_N slab wall neutron transport equation defined in Eqn. 1

$$\frac{1}{v_1}\frac{\partial\psi_1(x,t,\mu)}{\partial t} + \mu\frac{\partial\psi_1(x,t,\mu)}{\partial x} + \Sigma_1\psi_1(x,t,\mu) \\ = \frac{1}{2}\left(\Sigma_{s,1}\int_{-1}^{1}\psi_1(x,t,\mu)\partial\mu + \Sigma_{s,2\to1}\int_{-1}^{1}\psi_2(x,t,\mu)\partial\mu + Q_1\right) \tag{30a}$$

and

$$\frac{1}{v_2}\frac{\partial\psi_2(x,t,\mu)}{\partial t} + \mu\frac{\partial\psi_2(x,t,\mu)}{\partial x} + \Sigma_2\psi_2(x,t,\mu) \\ = \frac{1}{2}\left(\Sigma_{s,2}\int_{-1}^{1}\psi_2(x,t,\mu)\partial\mu + \Sigma_{s,1\to2}\int_{-1}^{1}\psi_1(x,t,\mu)\partial\mu + Q_2\right) \tag{30b}$$

which yields continuous functional representations for solutions a source term,

$$Q_1(x,t,\mu) = \frac{2}{v_1} + 2\mu + 2\Sigma_1(\mu+t+x) - \Sigma_{S,1}(2t+2x) - 2\Sigma_{S,2\to1}tx^2 \tag{31a}$$

and

$$Q_2(x,t,\mu) = \frac{2x^2}{v_2} + 4\mu tx + 2\Sigma_2(\mu+tx^2) + \Sigma_{S,1\to2}(2t+2x) - 2\sigma_{S,2}tx^2 \tag{31b}$$

Note that these equations are fully defined by independent variables and material data.

Our time-space discretization stencil defined in figure2 contains four quantities of interest. Left and right half-cell integrated time edge, and left and right half-cell integrated time-averaged source and angular flux terms. To get the expected solution back we must supply the numerical solver these specific values. Note that at this point the equations become quite long so include only thier dfinities in text. Full definitions can be found in the appendix A. We start by computing the left and right half-cell spatial average time-edge sources using a known x-location and known Δx,

$$Q_{1,L,j,k+1/2}(\mu) = \int_{x_{j-1/2}}^{x_j} Q_1(x,t=t_{k+1/2},\mu)\partial x \tag{32a}$$

and

$$Q_{1,R,j,k+1/2}(\mu) = \int_{x_{j-1/2}}^{x_j} Q_1(x,t=t_{k+1/2},\mu)\partial x \tag{32b}$$

then evaluate those time edge quantities over a time step

$$Q_{2,L,j,k+1/2}(\mu) = \int_{x_j}^{x_{+1/2}} Q_2(x, t = t_{k+1/2}, \mu) \partial x \tag{32c}$$

and

$$Q_{2,R,j,k+1/2}(\mu) = \int_{x_j}^{x_{j+1/2}} Q_2(x, t = t_{k+1/2}, \mu) \partial x \tag{32d}$$

for both groups. Finally initial conditions and boundary values of the angular flux must be defined. This is done with the original manufactured solution in equation 29. This is defined on the same stincle as the source shown in figure 2. Again these equations are shown in their full glory in appendix A. Figure blah shows the matching profile of our manufactured solution and that supplied from our discretization scheme. They match.

2.6. *Implementation in Code*

When implementing our iterative algorithms and discretization scheme for GPUs what to use as a LAPACK solver becomes an issue. Many large scale high performance open source liner algebra tools exist (e.g. Trillinos, PETSc, SUNDIALS, etc.) we chose a vendor supplied package depending on the hardware target of choice. We preform this performance analysis on a single GCD of MI250x. Thus we use the ROCm compute library to solve the system of equations. Modern GPU vendor supplied LAPACK libraries often include a `strided_batched` class of solvers. These will operate on a group of like-sized systems in unison and are optimized by the vendors of the hardware themselves. For example, LU decomposition with pivoting (the most generic direct solver for a system of linear equations) is implemented using RocSolvers `strided_batched_dgsev`.

Direct solvers where used in this work because all systems where small (with orders ranging between 4 and 1000). This makes the use of a strided batched implementation of LU decomposition with pivoting ideal. Furthermore in this work once that system is solved once the return of the A matrix is automatically L+U+D. In subsequent iterations this system can be backsolved very quickly For SI these kinds of optimizations do not performance as much as SI usually requires fewer iterations to converge a given problem. These optimizations will increase the memory footprint of OCI and SI as the A matrices are not usually stored. The only device kernels now required to make a whole itteration stay on the GPU in this scheme are the matrix builders which where already memory safe operations.

Profiling was done on both sets of itterative schemes kernels to ensure that

Both of the itterative schemes algrythims are presented in the next two sections

2.6.1. Source Itteration on GPU

Note that in thisthis case the A and b matrices are of dim If the functions are note created in the order established we must instead incur the functional imposition on the desired effort on the accelerator in the revers imposing

If the first itteration of

```
build $A$ in all cells
build constant part of $b$ in all cells
$l = 0$ ;                                                    //iteration counter
converged = false
while !converged do
    $\delta = \mathrm{SIZE}_{\mathrm{cellblocks}}$ ;                                //offset to a cell
    for $i$ = 0 to $N_{cell}$ ;                                   //transport sweep
     do
        if $l$=0 then
            //A in-out becomes the LU decomp
            //b in-out becomes the soultion vector
            strided batched dgesv from solver($A[\delta^2 i]$,$b[\delta * i]$)
        end
        else
            strided batched dgetrs($A[\delta^2 i]$,$b[\delta * i]$) ;       //back substitution
        end
        $e = ||b^l - b^{l-1}||_2$
        $\rho = e^l / e^{l-1}$
        if $e < tol(1 - \rho)$ then
            converged = true
        end
        $e^{l-1} = e^l$
        $l++$
    end
end
```

2.6.2. One Cell Inversion on GPU

The algorithms in OCI are much more sensitive to within iteration optimizations. This is because in many cases OCI will require a significantly larger number of iterations to converge the solution. For that reason it is imperative that the OCI iteration take place entirely on the GPU. The following algorithm was implemented to do just that with everything under the `while` loop wholly contained on the GPU.

```
build $A$ in all cells
build constant part of $b$ in all cells
$l = 0$ ;                                          //iteration counter
converged = false
while !converged do
    if l=0 then
        //A in-out becomes the LU decomp
        //b in-out becomes the soultion vector
        strided batched dgesv from solver($A$,$b$)
    end
    else
        strided batched dgetrs($A$,$b$) ;          //back substitution
    end
    $e = ||b^l - b^{l-1}||_2$ ;                    //L2 norm
    $\rho = e^l / e^{l-1}$ ;                       //spectral radius estimation
    if $e < tol(1 - \rho)$;                        //controlling for false convergence
    then
        converged = true
    end
    $e^{l-1} = e^l$
    $l++$
end
```

As in the source iteration algorithm the the systems are assumed dense. They are built in a strided batched configuration as to take advantage of the block sparsity. However this time all systems can be solved in unison without needing to be redispatched for every. The parallel solvers can solve for all cells and angles at once which cannot be done in source iterations. This OCI algorithm full allows the solvers provided from RocSolver decide on the most advantageous parallelism structure effectively off loading. The use makes no decisions about thread blocks. The only device kernels required are the matrix builders which are simple to build as they are already thread safe operations.

In higher dimensions this algorithm could remain the same and still be performant, again something that cannot be done traditional source iterations which requires KBA. In production codes the A systems are usually built on the fly as to avoid filling memory. This wouldn't nessacarioly preclude the use of these strided batch This work did not do that as One could imagine an algrythim that An additional consideration would have to be taken if an acceleration scheme is used in conjunction with this iterative scheme

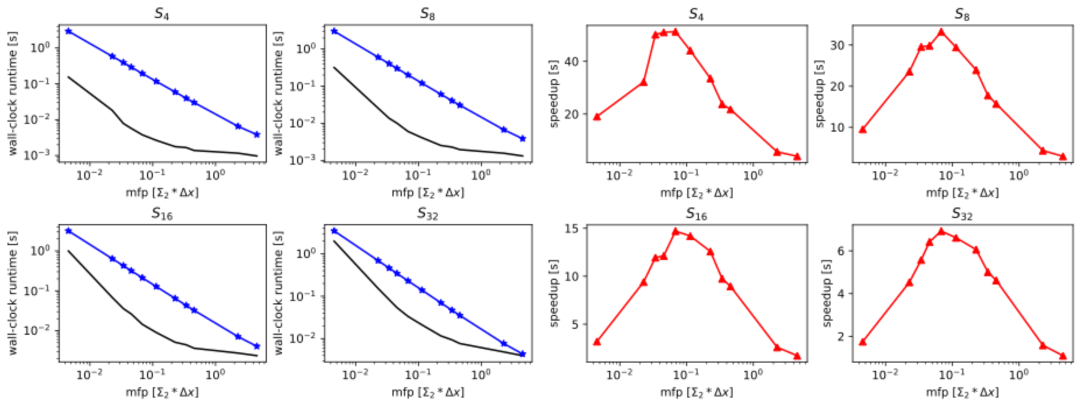

Figure 4. Wall clock runtime of OCI black and SI red AT LEFT and speedup AT RIGHT of Rosa, et. al. test problem over choice of mean free path in various quadrature orders

3. Performance Results

4. Time Dependent Impacts to Convergence Rates

To explore how time discretization impacts the convergence of OCI and SI we again start with the Rosa, et. al. test problem. Figure 5 shows how we vary the choice of mean free path and time step and compare iteration count of SI and OCI. SI expectedly has a constant number of iterations required between choices of mean free path. In other words the blue curve is the same in every plot. SI does improve convergence rate with respect to time discretization believed to be to improvements to scattering behavior. Also expectedly OCI improves with *both* larger choices of mfp and smaller times steps. OCI's convergence rate improvements are due to both improvements to scattering behavior as in SI and the postulated nature of smaller time steps being inversely correlated to cellular mfp. In fact the rate of improvement of the convergence rate actually actually increases in the thin limit. This confirms our hypotheses from observations of the spectral radius, that for smaller time steps OCI's converges more rapidly even for problems in the thin limit. If sufficiently small time steps are enabled this might mean a first order time discretization like backward Euler could be employed rather then the more expensive time dependent multiple balance

5. Conclusions, Discussion, and Future Work

In higher spatial dimensions we expect the relative performance of OCI to only grow with respect to SI. KBA algrythms are not well suited for GPUs where in-kernel dynamic work allocation is imposible. Thus kernels have to be launched with the OCI will better saturate the

For domain decomposed problems we expect OCI to out perform SI as the system can Domain decomposed SI requires communication of wave fronts

For highly scattering problems we expect that OCI will always underperform SI.

For problems on unstructured meshes we expect OCI will outperform SI.

Regardless of how performant an implementation of OCI is, the failure to converge problems in the thin-limit will continue to mean only lack-luster at best performance in some problems. In this paper we compared un-preconditioned SI to un-preconditioned OCI. When in production SI uses the has a well accepted set of acceler-

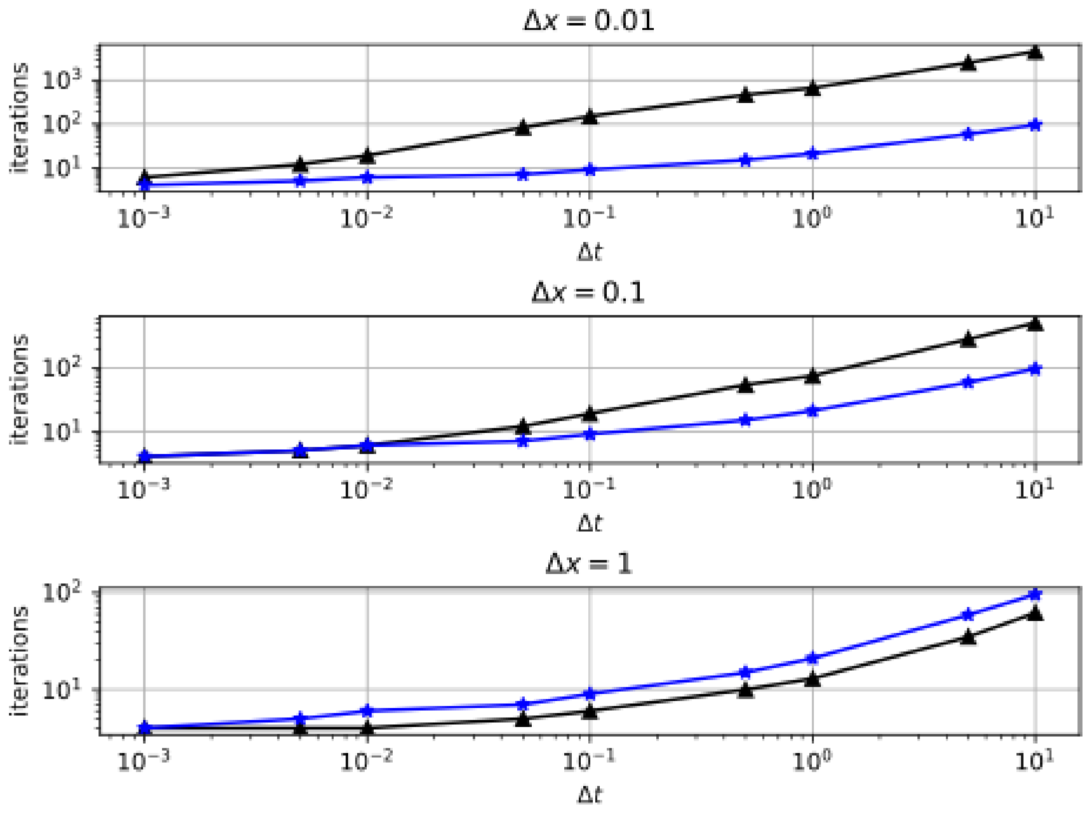

Figure 5. Time dsecritization thru

ation schemes/preconditioners (most popularly diffusion synthetic acceleration which aids in the diffusive limit, where SI's slowest converging error modes are). Likewise some acceleration/preconditioning scheme may exist that can converge OCI's slowest converging error mode: the thin-limit. Going forward we will explore acceleration schemes for OCI more rapidly converge the thin-limit and improve the overall perfoamcne.

Perofmance of vendor supplied libraries for a given matrix size

Another massive issue is memory foot print of OCI Sparsity could be exploited

This all copled with the ability to decrease the need for OCI

GMRES

Acknowledgement(s)

This work was supported by the Center for Exascale Monte-Carlo Neutron Transport (CEMeNT) a PSAAP-III project funded by the Department of Energy, grant number: DE-NA003967.

References

Adams, Marvin L. 1997. "Subcell balance methods for radiative transfer on arbitrary grids." *Transport Theory and Statistical Physics* 26 (4-5): 385–431. https://doi.org/10.1080/00411459708017924.

Anistratov, Dmitriy Y., and Yousry Y. Azmy. 2015. "Iterative stability analysis of spatial domain decomposition based on block Jacobi algorithm for the diamond-difference scheme." *Journal of Computational Physics* 297: 462–479. https://doi.org/10.1016/j.jcp.2015.05.033, Accessed 2024-04-18. `https://www.sciencedirect.com/science/article/pii/S0021999115003678`.

Baker, Randal S. 2017. "An SN Algorithm for Modern Architectures." *Nuclear science and engineering* 185 (1): 107–116.

Gleicher II, Frederick N., Yaqi Wang, Derek Gaston, and Richard Martineau. 2012. "The Method of Manufactured Solutions for RattleSnake, A SN Radiation Transport Solver Inside the MOOSE Framework." In *Transactions of the American Nuclear Society*, Vol. 106, Chicago, IL, June. American Nucelar Society.

Hoagland, Dylan S., and Yousry Y. Azmy. 2021. "Hybrid approaches for accelerated convergence of block-Jacobi iterative methods for solution of the neutron transport equation." *Journal of Computational Physics* 439: 110382. https://doi.org/10.1016/j.jcp.2021.110382, Accessed 2024-04-18. `https://www.sciencedirect.com/science/article/pii/S0021999121002771`.

Kunen, Adam J, Teresa S Bailey, and Peter N Brown. 2015. "KRIPKE - A MASSIVELY PARALLEL TRANSPORT MINI-APP." In *American Nuclear Society Joint International Conference on Mathematics and Computation, Supercomputing in Nuclear Applications, and the Monte Carlo Method*, April. `https://www.osti.gov/biblio/1229802`.

KWACK, JAEHYUK. 2022. "ROOFLINE PERFORMANCE ANALYSIS W/ INTEL ADVISOR ON INTEL CPUS & GPUS." In *2022 ECP ANNUAL MEETING*, `https://crd.lbl.gov/assets/Uploads/ECP22-Roofline-4-Intel-and-ALCF.pdf`.

Lewis, Elmer Eugene, and Warren F Miller. 1984. *Computational methods of neutron transport.* New York, NY, USA: John Wiley and Sons, Inc.

Meurer, Aaron, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, et al. 2017. "SymPy: symbolic computing in Python." *PeerJ Computer Science* 3: e103. https://doi.org/10.7717/peerj-cs.103, `https://doi.org/10.7717/peerj-cs.103`.

Morgan, Joanna Piper, Ilham Variansyah, Todd S. Palmer, and Kyle E. Niemeyer. 2023. "Exploring One-Cell Inversion Method for Transient Transport on GPU." In *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering*, Niagra Falls, CA, 8. CNS. `https://www.doi.org/10.48550/arXiv:2305.13555`.

Rosa, Massimiliano, James S. Warsa, and Michael Perks. 2013. "A Cellwise Block-Gauss-Seidel Iterative Method for Multigroup SN Transport on a Hybrid Parallel Computer Architecture." *Nuclear Science and Engineering* 174 (3): 209–226. Publisher: Taylor & Francis _eprint: https://doi.org/10.13182/NSE12-57, https://doi.org/10.13182/NSE12-57, Accessed 2024-04-18. `https://doi.org/10.13182/NSE12-57`.

Roy, Christopher J. 2005. "Review of code and solution verification procedures for computational simulation." *Journal of Computational Physics* 205 (1): 131–156. https://doi.org/10.1016/j.jcp.2004.10.036, Accessed 2024-06-26. `https://linkinghub.elsevier.com/retrieve/pii/S0021999104004619`.

Schunert, Sebastian, Cormac T. Garvey, Raffi Yessayan, and Yousry Y. Azmy. 2018. "ANALYSIS OF COMMUNICATION PERFORMANCE DEGRADATION OF THE RADIATION TRANSPORT CODE PIDOTS ON HIGH-UTILIZATION, MULTI-USER HPC SYSTEMS." In *PHYSOR: Physics of Reactors 2018*, Cancun, Mexico, 4. ANS. `https://www.osti.gov/biblio/1478365`.

Tramm, John R, and Andrew R Siegel. 2021. "Domain Decomposed Random Ray Neutron Transport on GPU-Based Systems." In *International Conference on Mathematics and Com-*

putational Methods Applied to Nuclear Science and Engineering (M&C 2021), .

Variansyah, Ilham, Edward W. Larsen, and William R. Martin. 2021. “A ROBUST SECOND-ORDER MULTIPLE BALANCE METHOD FOR TIME-DEPENDENT NEUTRON TRANSPORT SIMULATIONS.” *EPJ Web of Conferences* 247: 03024. https://doi.org/10.1051/epjconf/202124703024.

Wang, Jipu, William Martin, and Benjamin Collins. 2018. “The application of Method of Manufactured Solutions to method of characteristics in planar geometry.” *Annals of Nuclear Energy* 121: 295–304. https://doi.org/10.1016/j.anucene.2018.07.041, Accessed 2024-06-26. `https://www.sciencedirect.com/science/article/pii/S0306454918304031`.

Wolfe, Noah, and Xiaomin Lu. 2022. “Hierarchical Roofline on AMD Instinct™ MI200 GPUs.” In *Oak Ridge Leadership Computing Facility Traning Seminar*, `https://www.olcf.ornl.gov/wp-content/uploads/AMD_Hierarchical_Roofline_ORNL_10-12-22.pdf`.

Appendix A. Full MMS Source terms