International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)
Rio de Janeiro, RJ, Brazil, May 8-12, 2011, on CD-ROM, Latin American Section (LAS) / American Nuclear Society (ANS)
ISBN 978-85-63688-00-2

IMPROVED PARALLEL SOLUTION TECHNIQUES FOR THE INTEGRAL TRANSPORT MATRIX METHOD

R. Joseph Zerr
Department of Mechanical and Nuclear Engineering
The Pennsylvania State University
138 Reber Building, University Park, PA, USA
rjz116@psu.edu

Yousry Y. Azmy
Department of Nuclear Engineering
North Carolina State University
1110 Burlington Engineering Laboratories, Raleigh, NC, USA
yyazmy@ncsu.edu

ABSTRACT

Alternative solution strategies to the parallel block Jacobi (PBJ) method for the solution of the global problem with the integral transport matrix method operators have been designed and tested. The most straightforward improvement to the Jacobi iterative method is the Gauss-Seidel alternative. The parallel red-black Gauss-Seidel (PGS) algorithm can improve on the number of iterations and reduce work per iteration by applying an alternating red-black color-set to the sub-domains and assigning multiple sub-domains per processor. A parallel GMRES(*m*) method was implemented as an alternative to stationary iterations. Computational results show that the PGS method can improve on the PBJ method execution time by up to 10× when eight sub-domains per processor are used. However, compared to traditional source iterations with diffusion synthetic acceleration, it is still approximately an order of magnitude slower. The best-performing cases are optically thick because sub-domains decouple, yielding faster convergence. Further tests revealed that 64 sub-domains per processor was the best performing level of sub-domain division. An acceleration technique that improves the convergence rate would greatly improve the ITMM. The GMRES(*m*) method with a diagonal block preconditioner consumes approximately the same time as the PBJ solver but could be improved by an as yet undeveloped, more efficient preconditioner.

Key Words: transport, within-group, parallel, spatial decomposition, Gauss-Seidel, GMRES

1. INTRODUCTION

The rapid growth in computing platforms is creating a demand for computational methods that can fully utilize new architectures' resources in an efficient, scalable manner. In terms of neutron transport, distributing work based on energy or angular variable decomposition alone is inadequate when tens of thousands of processing elements (PEs) are available. This inadequacy is driven by the practical bounds to which the energy and angular variables are commonly refined. However, problems with ever-increasing numbers of spatial cells are highly desired to analyze physically larger regions and/or to employ finer spatial meshes. Therefore, focus has rightly shifted to decomposition in multiple variables with spatial domain decomposition of the within-group equations receiving the most attention.

For first-order, discrete ordinates neutron transport with a structured spatial mesh, the Koch-Baker-Alcouffe (KBA) [1], or wavefront, method has become a common choice for spatial domain decomposition of the within-group equations. For an n-dimensional (n = 2, 3) system, the domain is mapped to an $n-1$ processor topology. Mesh sweeps are performed in parallel over individual sub-domains by communicating angular fluxes on sub-domain boundaries to downstream neighbors, and advancing along diagonal planes. The KBA method sacrifices load balance in favor of a synchronous parallelization of the serial algorithm. The work is reordered only in the sense that it is distributed for faster execution, but the incoming and outgoing data managed by the multiple PEs is unchanged from their counterparts in the serial algorithm. The KBA method has been implemented with traditional source iterations (SI) in the PARTISN code [2], and more recently it has been used as an efficient way to perform matrix-vector multiplications for a GMRES(m) scheme in Denovo [3]. Furthermore, the KBA method has been implemented with diffusion synthetic acceleration (DSA) of the source iterations [2] and with a DSA-based preconditioner for the Krylov solver [3].

The integral transport matrix method (ITMM) [4–6] is designed to completely avoid repetitive mesh sweeps by formulating and storing operators that relate the scalar flux within a sub-domain to the angular flux on its boundaries. For a large problem, the global domain can be spatially decomposed into several sub-domains, [6] then the PE owning each sub-domain constructs its own ITMM operators and is tied to the rest of the global domain via angular fluxes at its boundaries. Therefore, the iterative process shifts from cell-average scalar fluxes, as in standard SI, to sub-domains' interfacial angular fluxes.

In this paper we present alternative solution methods to the parallel block Jacobi (PBJ) algorithm described in [6]. First, the parallel red-black Gauss-Seidel (PGS) was developed to split the sub-domains into a red/black pattern, alternating the computations of red and black sub-domains to use more up-to-date information. Second, the parallel GMRES (PGMRES) routine was tested as a means to abandon stationary iterations for a potentially more efficient Krylov subspace solver.

The remainder of this paper is organized as follows. Section 2 briefly outlines the ITMM, its operators, and the local-global solution strategy using PBJ. Then the PGS and PGMRES methods are described as alternatives to PBJ for the global solution. Section 3 presents a series of weak scaling test results. The PGS algorithm with increasing number of sub-domains is compared to PBJ. Then we show how the PGS algorithm performs compared to the KBA method as implemented in PARTISN. The PGS method is then scaled to much larger systems to determine how it behaves in massively parallel regimes. Weak scaling results for PGMRES are presented next to show where the method currently succeeds and highlight issues targeted for improvement. Section 4 provides a summary and the conclusions of our work.

2. PARALLELIZATION OF THE INTEGRAL TRANSPORT MATRIX METHOD

We have fully derived the ITMM process for a Cartesian mesh with diamond difference spatial discretization, discrete ordinates angular discretization, and isotropic scattering in [6]; the same set of equations will be applied here. Consider the within-group equations for some domain as an iterative process, using the previous iterate of the scalar flux vector to compute a new iterate.

With known incoming angular flux serving as sub-domain boundary conditions, the scalar flux can be computed with

$$\boldsymbol{\phi}^{(l+1)} = \boldsymbol{J}_{\phi}\left(\boldsymbol{\phi}^{(l)} + \boldsymbol{\Sigma}_s^{-1}\boldsymbol{q}\right) + \boldsymbol{K}_{\phi}\boldsymbol{\psi}_{in}. \quad (1)$$

Standard operator notation is applied. $\boldsymbol{\phi}$ is the scalar flux vector of length equal to the number of spatial cells N and is given a superscript to indicate the iteration index. $\boldsymbol{q}$ is the isotropic distributed source vector of length N. $\boldsymbol{\Sigma}_s$ is the $N\times N$ within-group diagonal scattering matrix. $\boldsymbol{\psi}_{in}$ is a vector of all incoming angular flux at the boundaries. Its length is half the number of cell-boundary surfaces N_b times the number of angles N_t. $\boldsymbol{J}_{\phi}$ and $\boldsymbol{K}_{\phi}$ are ITMM operators. $\boldsymbol{J}_{\phi}$ is an $N\times N$ matrix that represents the coupling among all sub-domain-cells' scalar fluxes; $\boldsymbol{K}_{\phi}$ is an $N\times N_bN_t$ matrix that represents the contribution the boundary angular fluxes make, after attenuation, to the sub-domain-cells' scalar fluxes. Upon convergence of Eq. (1), successive iterates of the scalar flux are equivalent and the system of equations can be rewritten as

$$\left(\boldsymbol{I} - \boldsymbol{J}_{\phi}\right)\boldsymbol{\phi} = \boldsymbol{J}_{\phi}\boldsymbol{\Sigma}_s^{-1}\boldsymbol{q} + \boldsymbol{K}_{\phi}\boldsymbol{\psi}_{in}. \quad (2)$$

The matrix $(\boldsymbol{I} - \boldsymbol{J}_{\phi})$ is called the integral transport matrix. [4]

For the purpose of spatial decomposition into sub-domains, we must be able to calculate the outgoing angular flux from a region, such that it can be used in the global iterative solution strategy. Continuing with the concept of full coupling in a region, the outgoing angular flux is computed with

$$\boldsymbol{\psi}_{out} = \boldsymbol{J}_{\psi}(\boldsymbol{\phi} + \boldsymbol{\Sigma}_s^{-1}\boldsymbol{q}) + \boldsymbol{K}_{\psi}\boldsymbol{\psi}_{in}. \quad (3)$$

The $\boldsymbol{J}_{\psi}$ operator couples cells' scalar flux to outgoing angular flux at the boundaries; it is an $N_bN_t\times N$ matrix. The $\boldsymbol{K}_{\psi}$ operator represents the attenuation of incoming angular flux of along a discrete ordinate from the incoming surfaces to the outgoing surfaces; it is an $N_bN_t\times N_bN_t$ matrix.

The ITMM operators are large and dense, so the size of systems that can be solved sequentially is very limited. To consider larger problems, parallel computing is necessary. Moreover, we have developed this method with the intention for deployment on massively parallel systems: minimally several hundreds of PEs but preferably thousands or tens of thousands of PEs.

When solving in parallel, the overall region is decomposed into sub-domains arranged in a 3-D mesh virtual process topology. Each sub-domain is treated as an independent problem; each has its own set of ITMM operators and boundary conditions. The scalar flux is computed for all cells in the sub-domain by using LAPACK routines to factorize $(\boldsymbol{I} - \boldsymbol{J}_{\phi})$ and solve Eq. (2). Using this information and the boundary conditions, all the outgoing angular fluxes are computed at the boundaries of the sub-domain with Eq. (3). The outgoing angular flux is passed between adjacent sub-domains in an iterative fashion. The exchanged data comprises a set of boundary conditions for a new calculation to compute the updated scalar flux distribution within the sub-domain. An iterative process takes place until convergence of the scalar flux in all sub-domains is achieved. This algorithm is in the mold of the spatial domain decomposition originally proposed in [7].

This iterative procedure is *not* an inner iteration scheme because it does not iterate on the scattering source. The critical aspect of this iterative method is that the quantities being iterated on are the angular fluxes at sub-domain boundaries, not the scalar fluxes within the sub-domain. The scalar fluxes serve as a computationally inexpensive, intermediate step in the computation of updated outgoing boundary angular fluxes and as a convenient tool for determining the convergence of the global system.

When each PE owns a single sub-domain, this iterative process reduces to the PBJ global solution method where the global problem is written as a single global system of equations. The nonzero elements of the resulting block coefficient matrix are the ITMM operators of the individual sub-domains. Scalar and angular fluxes, distributed among the participating processors, comprise the solution vector. The right hand side (RHS) vector, also distributed, contains the distributed sources and boundary conditions. A single iteration updates all the values of $\boldsymbol{\phi}$ and $\boldsymbol{\psi}_{out}$ before starting a new iteration, hence the "block Jacobi" terminology.

The most straightforward improvement to the Jacobi iterative method is the Gauss-Seidel alternative, wherein one uses the most current information when computing new values. In the PBJ algorithm the angular flux always lags because no improved information exists until communication. The parallel red-black Gauss-Seidel (PGS) algorithm can improve on the number of iterations by applying an alternating red-black color-set to the sub-domains. In the first half of a PGS iteration, the red sub-domains are solved for $\boldsymbol{\phi}$ and $\boldsymbol{\psi}_{out}$ using $\boldsymbol{\psi}_{in}$ from the previous black iteration. This is followed by sending the just-computed outgoing angular fluxes to adjacent black sub-domains. The PGS iteration is completed when black sub-domains use these incoming angular fluxes to compute a new iterate of their own $\boldsymbol{\phi}$ and $\boldsymbol{\psi}_{out}$.

To avoid the problem of processor idleness, PEs are given ownership of multiple sub-domains, half red and half black. Angular fluxes at sub-domain boundaries are exchanged either via communication over the network interconnect or via vector copying within an individual PE's memory space. PEs are assigned at least two sub-domains to eliminate idleness when alternating between the red and black halves of the iteration.

We have previously found with PBJ that increasing the number of sub-domains will increase the number of PBJ iterations. [6] Yet when a modest increase in the number of sub-domains is coupled with the PGS implementation, the number of iterations will only grow slightly and in some cases decrease because of the faster rate of convergence with PGS. Each further division of a sub-domain into even smaller ones will increase the number of PGS global iterations necessary for convergence. Nevertheless, consider the aforementioned sizes of the ITMM operators. The smaller sub-domains provide superlinear benefits in required computational resources (memory and time) during construction and iterative application of the ITMM operators. Therefore, our work has focused on optimizing the run-time based on these competing factors.

As an alternative to the stationary iterative methods, PBJ and PGS, a potentially more efficient Krylov subspace solver can be employed for the global solution. The global problem's coefficient matrix is not symmetric, and we can say nothing generally about its positive definiteness. With these characteristics, the GMRES algorithm is the only solver robust enough for implementation. GMRES requires matrix-vector multiplications to build the subspace that is

used to minimize the residual and compute updates to the solution vector. [8] These multiplications manifest themselves simply as ITMM operators multiplying the residuals of $\boldsymbol{\phi}$ and $\boldsymbol{\psi}_{out}$ and the basis vectors of the same sub-domain and are therefore already distributed in our PGMRES algorithm. Communications with adjacent sub-domains are used to exchange sub-vectors of the residual and orthogonal basis vectors. Global communications are necessary to perform reductions on the data needed to form the least squares problem.

Because of the large size of the global problem, the PGMRES method was implemented using *m* restarts to limit the size of the subspace being built. This has the potential to cause stagnation, stalling the reduction in the error if the restart number is set too low. To improve the method, a diagonal block preconditioner is employed. [8] The benefit of this preconditioner is that the diagonal blocks of the coefficient matrix are $(\boldsymbol{I}-\boldsymbol{J}_{\phi})$ for the $\boldsymbol{\phi}$ equations and $\boldsymbol{I}$ for the $\boldsymbol{\psi}_{out}$ equations. No additional operations are necessary to build the preconditioner except to factorize $(\boldsymbol{I}-\boldsymbol{J}_{\phi})$. The PGMRES method increases the work per iteration compared to the stationary methods. However, we seek to reduce the number of iterations sufficiently that the overall execution time decreases.

3. COMPUTATIONAL RESULTS

Several tests have been performed to analyze the weak scaling of our parallel ITMM code. Previously, strong scaling results were presented in [6]. In weak scaling tests, the size of the problem is increased with the number of processors. Specifically, we maintain a fixed number of spatial cells per processor, holding the number of unknowns constant for a given number of sub-domains per processor. However, changing the number of sub-domains per processor for the PGS approach will change the number of $\boldsymbol{\psi}$ unknowns. We are interested in the growth in the number of iterations and execution time as the number of processors, and therefore sub-domains, is increased.

For the moment, consider a single sub-domain per processor. We rely on a generic model problem and modify cell dimensions and material scattering cross sections to investigate the effects optical thickness and scattering ratio have on our method. The base model domain is a cube with side length *L* and four materials, as shown in Fig. 1, having no symmetries that may influence iterative convergence. The domain is discretized via a uniform mesh of cubic cells and the number of cells in the base model is scaled, changing the size of the ITMM operators.

For the weak scaling tests, additional processes are added to the 3-D virtual topology, and the global domain is divided into sub-domains of equal size. The sub-domains are not periodic repetitions of the base; instead, the base model domain is stretched in the dimension(s) that receives additional PEs and divided into cubic sub-domains.

When each PE is assigned multiple sub-domains, the base model itself is divided into equally sized, cubic sub-domains, half red and half black. As the problem is scaled to more PEs, the base model is distorted in the same manner described above: each additional PE gets a cubic region, which it divides into equally sized cubic sub-domains.

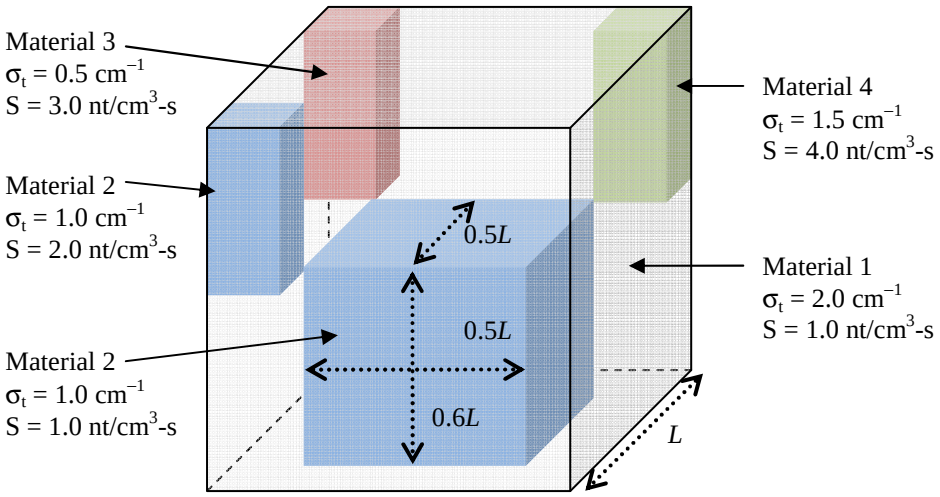

Figure 1. Test cases' base model of four materials and four source strengths.

Our code has been written in the Fortran 90/95 standard. To achieve parallelism for targeted distributed memory architectures, we used the Message Passing Interface (MPI) instruction set. Standard compiler optimizations were utilized for improved performance on each PE, namely in the vectorization of matrix-vector operations. All cases were run multiple times, and we report the average execution times for a more accurate estimate of typical system performance.

Test cases were performed on three different computer systems. At Los Alamos National Laboratory (LANL), we used the Yellowrail (YR) network [9] to test our methods on up to P = 256, where P is the number of PEs. YR is a distributed memory cluster of 139 nodes with 8 CPUs (1 CPU = 1 PE) and 16 GB of memory per node. Nodes are organized into a single Connected Unit (CU), whereby a single, large switch connects all the nodes. With the initial observations, we then scaled cases to P = 1,024 on LANL's Redtail (RT) system [9]. RT is composed of 14 CUs. Each RT CU has 131 nodes with 8 CPUs and 32 GB of memory per node. For access to thousands of PEs to test our code, we used the Cray XT5 JaguarPF (CXT5) cluster [10] at Oak Ridge National Laboratory (ORNL). CXT5 is composed of 18,688 nodes, each with 12 CPUs, 16 GB of memory and a SeaStar 2+ router. The interconnect is a 3-D torus topology.

3.1. Comparing PGS and PBJ Performance

Comparing the PGS and PBJ technique involves evaluating the relative performance amid competing effects. The superlinear reduction in operator sizes leads to smaller memory requirement per PE, faster ITMM construction time, and faster matrix-vector operations for PGS in spite of the fact that each PE handles multiple sub-domains. Conversely, per the results of [7] and [6], we know that for a fixed problem, increasing the number of sub-domains will induce an increase in the number of iterations for convergence. Moreover, the increase in the number of sub-domains per PE requires an increase in the number of communications to other PEs as well as substantial copying of arrays among sub-domains on the same PE.

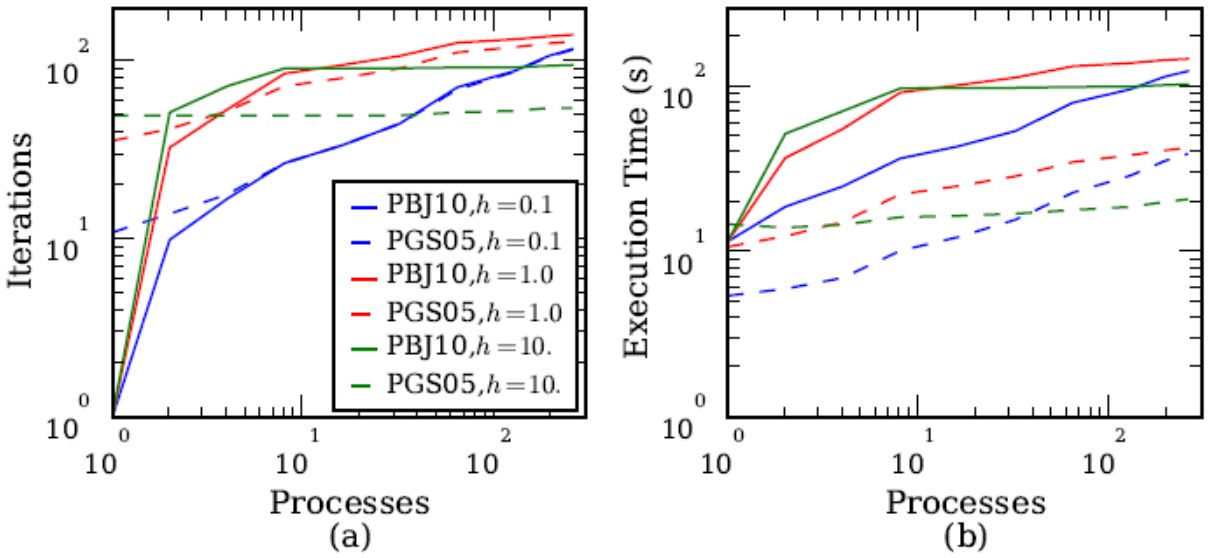

Figure 2. PBJ/PGS test cases on YR with S_{16}, 10×10×10 base model, c = 0.99, & varying h.

We start with a problem where the base model is 10×10×10 spatial cells and the angular quadrature order is 16 in the LQ_N quadrature set. (We will continue to use the LQ_N set for the remainder of this paper.) Additional PEs are added in one dimension at a time, and we cycle through the dimensions to maintain a low global surface-to-volume ratio. The PBJ case therefore assigns a single 10×10×10 sub-domain to each PE, and the PGS case assigns eight 5×5×5 sub-domains to each PE.

Tests were performed on YR up to P = 256, with scattering ratio c = 0.99, and considering cell dimension h = 0.1, 1.0, and 10.0 cm to model the effects of varying the optical thickness of the host media. (Alternatively, the total interaction cross section could have been scaled to achieve the same result.) In Fig. 2a, the number of iterations *vs* P is shown for the six cases. The three PBJ cases (solid lines) require only a single iteration for P = 1 because the entire problem consists of that single (sub-)domain. The PGS cases (dashed lines), in contrast, require multiple iterations. However, as P increases the two methods have very similar trends in the growth of iterations, and in the optically thick problem PGS uses significantly fewer iterations than PBJ. The total execution time is shown in Fig. 2b, and one can clearly see the benefit of the PGS decomposition: faster operation times and lower memory burden coupled with red-black iterations reduce execution time considerably, nearly a full decade in the h = 10.0 cm case.

We repeated the experiment but used a lower order quadrature, S_4, and increased the number of cells of the base model to 16×16×16. For PBJ, sub-domains of this size require most of the memory available per PE. PGS cases have eight 8×8×8 cells per PE. Tests were performed on YR up to P = 256. Figure 3a shows that the iteration counts are still slightly better for the PGS case with eight sub-domains per PE. Moreover, PGS continues to outperform PBJ in execution time by a significant factor.

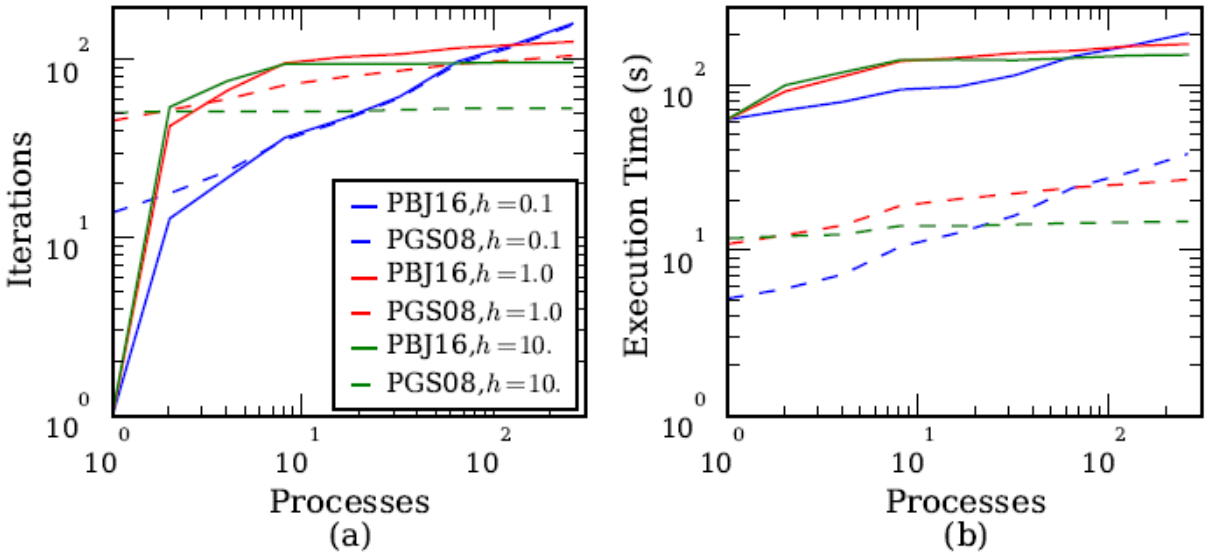

Figure 3. PBJ/PGS test cases on YR with S_4, 16×16×16 base model, c = 0.99, & varying h.

Although the results presented thus far show a decrease in the number of PGS iterations from PBJ iterations, one should not assume that this trend will continue with increasing number of red/black sub-domain divisions. Further divisions of the sub-domains for the PGS approach will lead to higher iteration counts, and the tradeoff between the decreasing work per iteration and the increasing number of iterations will be highlighted more clearly via tests presented shortly.

3.2. Comparing PGS and KBA Performance

Evident from the results presented in the previous subsection, PGS is a more attractive solution method than PBJ, at least for a modest number of sub-domains per PE. We now test the ITMM with PGS against the KBA method for mesh sweep parallelization. We have run similar test cases as above using our code with the PGS method and compare the results to the same test cases using PARTISN with the KBA method. PARTISN solves the within-group equations with the SI scheme and uses diffusion synthetic acceleration (SI DSA) to improve the convergence rate. The ITMM code currently does not include acceleration or preconditioning of the global problem, and improvement of the convergence rate remains an open challenge which we discuss further in subsequent sections.

We wish to run a larger, more practical problem that will fully utilize the available memory when run with the PGS method, not the PBJ method. Therefore, we use the 16×16×16 base model with eight 8×8×8 sub-domains per PE and increase the quadrature order to S_8. The scattering ratio is still c = 0.99, and the cell cubic dimension is again varied as h = 0.1, 1.0, and 10.0 cm. Additional PEs are added in the z- and y-directions before the x-direction to maximize PARTISN's parallel efficiency—i.e., the 2-D spatial decomposition for KBA is done in the y-z plane. Tests are performed on the RT system up to P = 1,024.

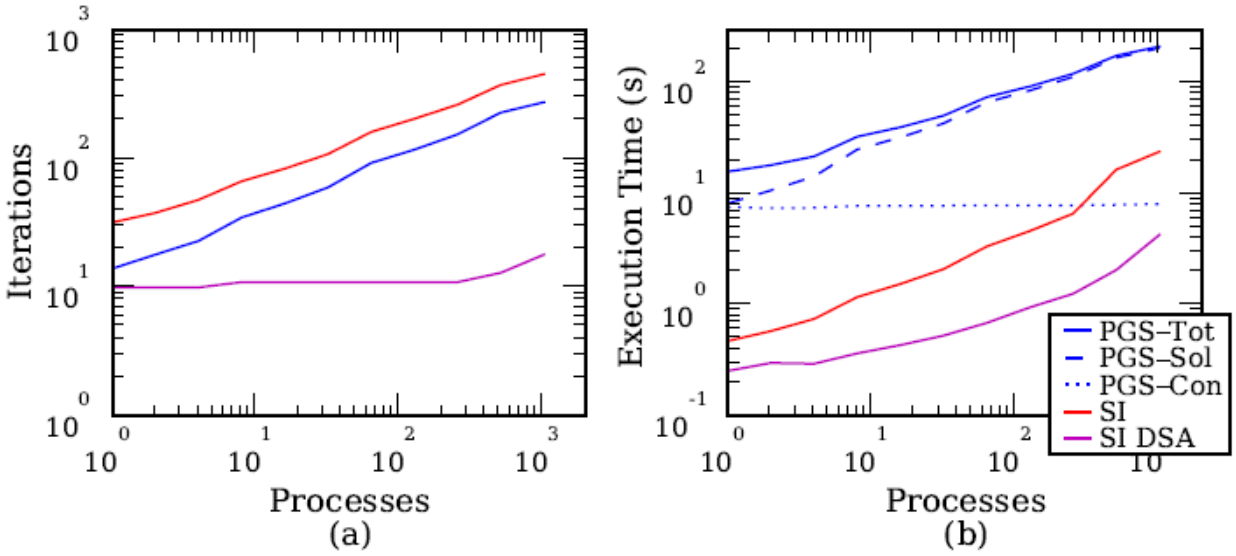

Figure 4. PGS/KBA test case on RT with S_8, 16×16×16 base model, c = 0.99, & h = 0.1 cm.

Figure 4 shows the results for c = 0.99 and h = 0.1 cm. It must be noted that the work per iteration is very different for the three methods presented: PGS, SI, and SI DSA. We use these plots of iteration counts to observe trends in the amount of work to be performed per PE per method. For the present case the PGS and SI cases show a quickly growing number of iterations with increasing P in Fig. 4a; with the high scattering ratio, cells are tightly coupled because the absorption loss mechanism is weakened. Yet SI DSA convergence is unsurprisingly rapid and the curve is flat except for a slight increase at P = 1,024; the acceleration scheme quickly kills diffusive error modes and the small optical thickness makes for a so-called “leaky” system, handled well by the transport sweeps. [11] The PGS execution time, shown in Fig. 4b, is divided into the ITMM operator construction time (dotted line), iterative solution time (dashed line), and the total time, a sum of those two components (solid line). Construction time is essentially constant because the number of sub-domains per PE and their sizes are constant. Up to P = 1,024, both SI and SI DSA outperform the PGS method by a large factor. In fact, the ITMM operator construction time exceeds the entire SI DSA time. However, the SI times are growing slightly faster, needing to perform a mesh sweep every iteration. Moreover, every time the domain is expanded in the x-direction, the efficiency of the KBA method is diminished, most noticeable by a change in the slope of the SI execution time line from P = 256 to 512.

The results for the c = 0.99, h = 1.0 cm case are given in Fig. 5. Now the number of iterations is relatively flat for all methods. The PGS method has a very slight growth, indicating that the sub-domains are not yet fully decoupled and some effects are not completely localized. The increasing size of the system has little effect on the necessary number of source iterations for convergence. The exceptional performance of SI DSA leads to it having a much faster execution time than both PGS and SI. The large number of source iterations raises its execution time, bringing it close to PGS for large P. The PGS method execution time grows slowly due to the slow growth in iteration count.

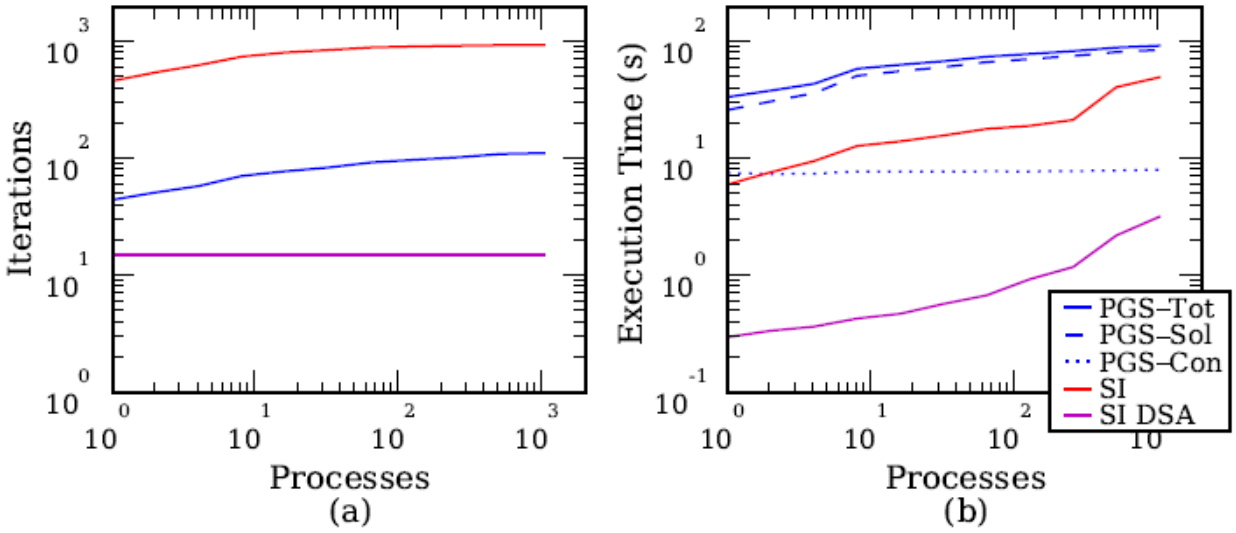

Figure 5. PGS/KBA test case on RT with S_8, 16×16×16 base model, c = 0.99, & h = 1.0 cm.

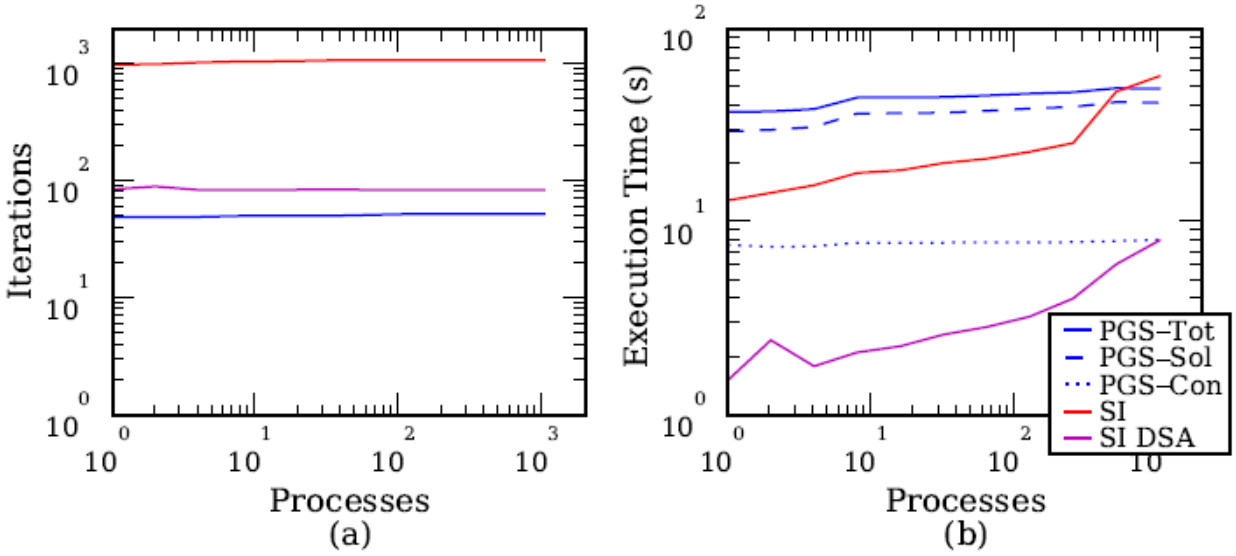

Figure 6. PGS/KBA test case on RT with S_8, 16×16×16 base model, c = 0.99, & h = 10.0 cm.

The c = 0.99, h = 10.0 cm case is the most optically thick case considered. Increasing optical thickness most benefits the PGS method because the sub-domains become decoupled—transport effects are more localized and fewer iterations are needed. This is the toughest case for SI and SI DSA. Although their respective iteration curves shown in Fig. 6a are flat, they are both considerably higher than the previous cases considered. The number of PGS iterations is below SI and SI DSA, but the greater amount of work per iteration still results in the longer execution times observed in Fig. 6b. However, the large number of source iterations results in a longer SI execution time than PGS for P = 1,024, caused partly by the jump in SI execution time at P =

512. The PGS execution time grows much more slowly than SI and SI DSA due to the flat growth in number of iterations of decoupled sub-domains.

The results thus far delineate circumstances in which the ITMM with PGS will be competitive with SI and SI DSA. Namely, problems with large optical thickness undergo a decoupling effect that provides PGS an advantage to converge more quickly. The comparison tests with PARTISN reveal that the ITMM with PGS works well when the sub-domains are optically thick, keeping transport effects and sub-domains decoupled. This results in a slow growth in the number of iterations for convergence and nearly flat growth in total execution time. Furthermore, as the problem is scaled to a greater number of cells, the sweeps become increasingly expensive, whereas the ITMM sub-domain sizes remain constant. The parallel solution times of SI and SI DSA continue to grow even when the iteration curves are flat because the size of the domain to be swept continues to grow. Even for the h = 1.0 cm cases, the faster growth in execution time for SI and SI DSA suggests superior scalability for PGS and potentially faster execution time when P is increased to 10,000+ PEs. Moreover, increasing the number of sub-domains per processor for PGS can lead to improved execution times, as illustrated in the following subsection. However, the large difference in PGS and SI DSA execution times suggests that PGS global iterations must be accelerated to increase the convergence rate as DSA does for SI.

3.3. Comparing Levels of PGS Sub-Domain Division in the Massively Parallel Regime

Increasing the level of PGS sub-domain division simply requires successive division of a single sub-domain on a PE (PBJ) into more, smaller sub-domains. The results so far have considered a single division in each dimension to yield eight sub-domains per processor. Continuing with the base model used in Sec. 3.2, we consider cases where the PE is assigned eight 8×8×8 (PGS08), 64 4×4×4 (PGS04), and 512 2×2×2 (PGS02) sub-domains. Test cases use an S_8 quadrature set, c = 0.99, and h = 0.1, 1.0, and 10.0 cm. To test the PGS method on a more massively parallel regime than was accessible on the LANL systems, we performed these calculations on the CXT5 system at ORNL. To fit the problems on the nodes, in terms of memory, we restricted runs to eight PEs per node instead of using all twelve. Computations were performed up to P = 32,768.

The results for the h = 0.1 cm case are presented in Fig. 7. All three levels of PGS division follow the same trend in increasing number of iterations. The curves are shifted upward for increasing number of sub-domains as expected. The total execution time for low P shows that the smaller sub-domains perform better, likely caused by the faster operations performed per iteration and the reduced time for ITMM operators' construction. However, for large problems, the construction time becomes increasingly unimportant in the total execution time. The larger number of iterations burdens the communication network and the PGS02 case eclipses PGS04 and PGS08 in execution time. The PGS04 execution time steadily remains below the PGS08 execution time. These two cases show similar curves with PGS04 having a slightly faster growth than PGS08, especially as P is increased to thousands of PEs.

The small cell-optical-thickness results in Fig. 7 are expected given the tight coupling among the sub-domains. Increasing the optical thickness, shown in Fig. 8, flattens the curves in both the iterations and execution time plots. The same trends as the h = 0.1 cm case are present, but the

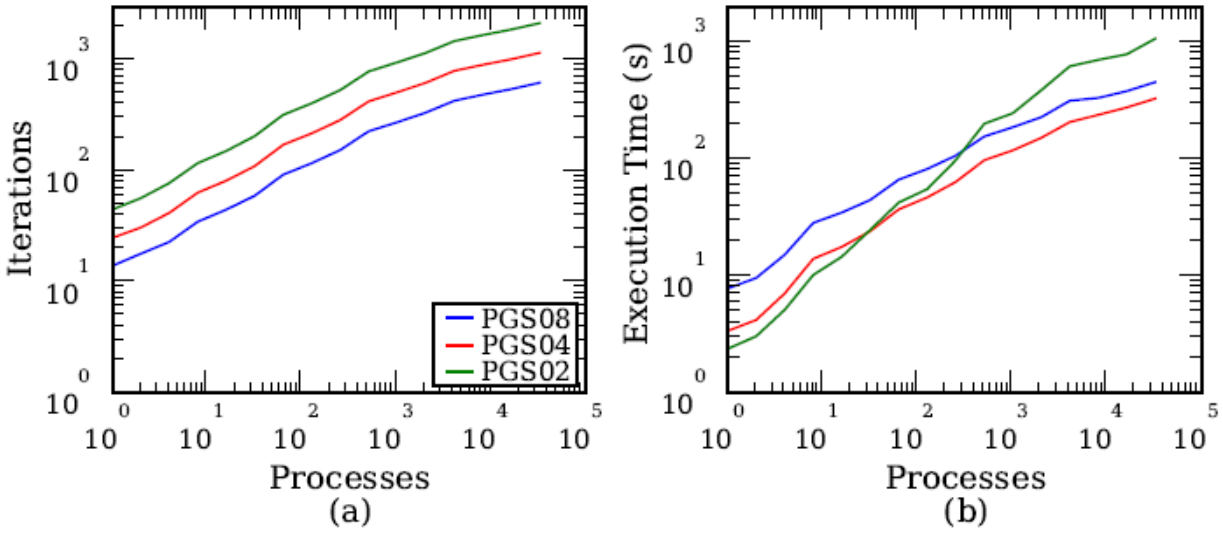

Figure 7. PGS test cases on CXT5 with S_8, 16×16×16 base model, c = 0.99, & h = 0.1 cm.

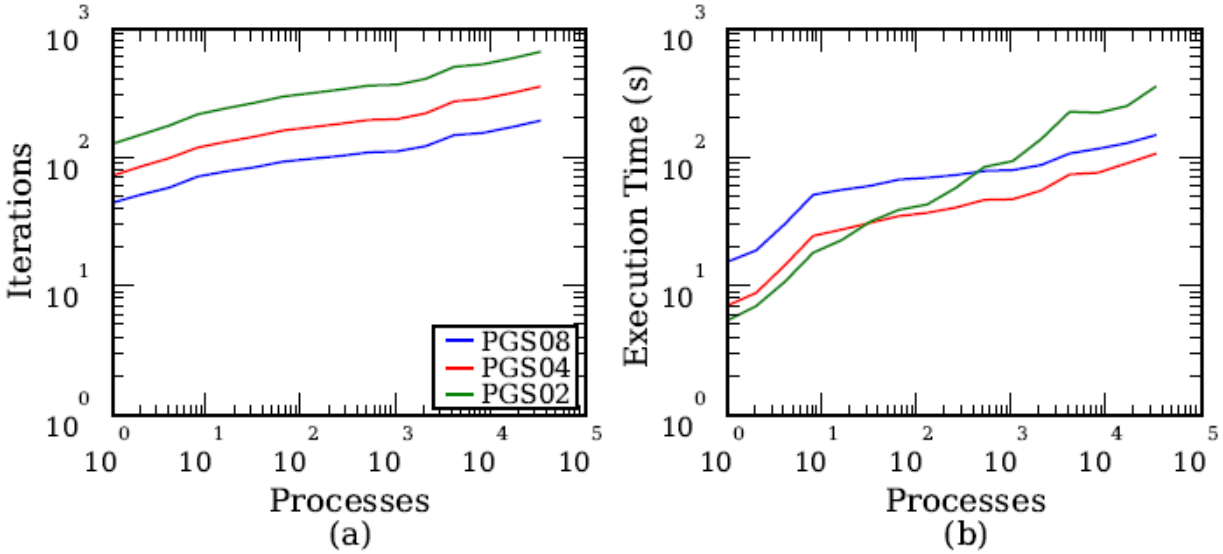

Figure 8. PGS test cases on CXT5 with S_8, 16×16×16 base model, c = 0.99, & h = 1.0 cm.

growing execution time is much slower and the method presents much better scaling properties. Again, the PGS04 case is the fastest PGS division.

The optically thick, h = 10.0 cm cases achieve excellent scalability for the PGS08 and PGS04 strategies. Shown in Fig. 9 the iteration curves are nearly completely flat, suggesting rapid decoupling of the sub-domains and fast convergence. The execution time plots continue to show PGS04 as a slightly more efficient PGS division than PGS08, executing noticeably faster with only a slightly faster rate of growth.

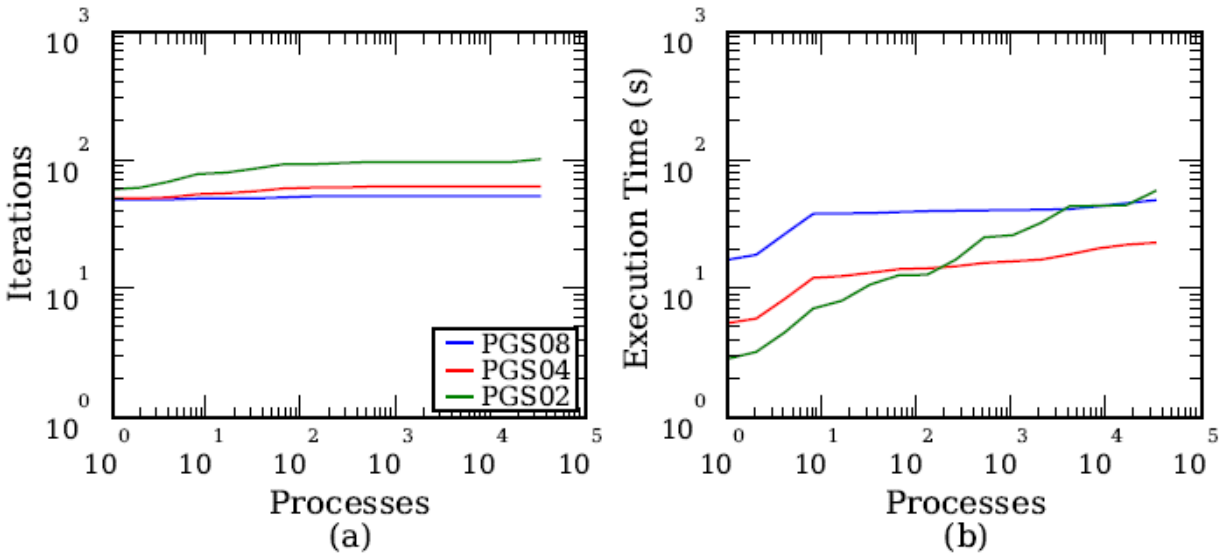

Figure 9. PGS test cases on CXT5 with S_8, 16×16×16 base model, c = 0.99, & h = 10.0 cm.

The results presented in this section clearly indicate that the PGS08 and PGS04 cases scale very similarly. The fact that PGS04 does not rapidly climb in execution time, like PGS02, makes it a very good choice beyond P = 32,768. More generally, the results suggest a single division in each dimension, like PGS08, is not necessarily the most efficient approach, even if it does mean similar or fewer global iterations than PBJ, as shown in Sec. 3.1. Moreover, the number of iterations and burden it places on the system for fast communications is the most influential factor in the fast growth of PGS02 execution time. If a method is developed to accelerate the ITMM with PGS iterations, more, smaller sub-domains may be the more attractive choice because of faster operator construction time and matrix-vector operations.

3.4. Comparing PBJ and PGMRES Performance

As mentioned in Sec. 1, source iterations can be replaced with Krylov subspace methods, and specifically the KBA parallel mesh sweep is used to perform matrix-vector multiplications necessary for the GMRES algorithm. The ITMM with spatial sub-domains can also be solved with a parallel GMRES method as an alternative to the PBJ and PGS stationary methods.

We return to the 10×10×10, S_{16} base model problem to compare the PBJ method with the PGMRES(m). For our PGMRES(m) calculations, we employed classical Gram-Schmidt orthogonalization, a diagonal block preconditioner, and m = 20. Running on the YR system up to P = 256, the results for varying h and c = 0.99 are given in Fig. 10. Evident in Fig. 10a the number of iterations for the two methods have very similar trends. The preconditioner helps keep the number of PGMRES iterations from growing too quickly, especially with a small restart number. Unfortunately, because the ITMM operators are large, a low restart number must be selected to avoid overflowing memory. The execution-time plots in Fig. 10b demonstrate generally that the two methods have similar execution time, with PBJ doing better for the optically thick problem, and PGMRES doing better for the optically thin problem.

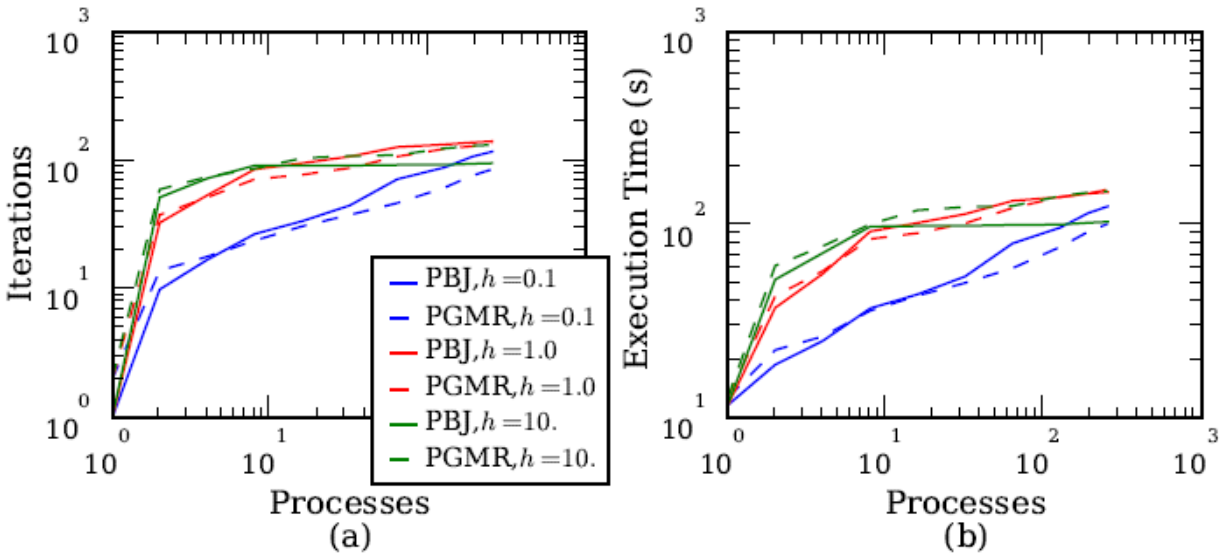

Figure 10. PGMRES/PBJ test cases on YR with S_{16}, 10×10×10 base model, m = 20, c = 0.99, & varying h.

PGMRES should benefit from two improvements. First, dividing single sub-domains and assigning a PE multiple sub-domains will permit faster matrix-vector operations without any significant changes to the parallel GMRES scheme. Second, a more powerful preconditioner can further improve the convergence rate of PGMRES and potentially reduce the execution time.

4. CONCLUSIONS

Two new methods for parallelizing the global ITMM solution have been introduced. The PGS method has shown an immediate benefit, reducing the execution time compared to PBJ significantly. This is driven by superlinear reductions in matrix-vector operations and the faster convergence rate Gauss-Seidel methods generally have compared to Jacobi methods. Comparing our PGS methods to SI and SI DSA indicates that PGS is most competitive for optically thick and highly scattering cases, where the SI approach exhibits slower convergence. Although the SI execution time seems to be growing faster than PGS, the SI DSA execution time is growing fairly slowly and remains at least a factor of ten faster than our method, even in the modestly massively parallel regime. Some of this gap can be closed by increasing the number of PGS sub-divisions, leading to shorter operations without an excessive gain in iterations. However, some acceleration technique, perhaps multigrid, will ultimately be necessary to improve the convergence rate of the PGS approach to make it generally more competitive.

The PGMRES method has been introduced as a second alternative to the PBJ method. Although results indicate similar performance to PBJ for test cases, the preconditioner selected was very simple. Potentially a more efficient preconditioner can be constructed to improve the convergence rate. This remains an open area of research; the preconditioner must be powerful, but it must be built cheaply to avoid the scenario where its construction begins to significantly add to total execution time. Moreover, Krylov solvers are gaining popularity for use in transport

codes because of their parallel scalability and robustness. Our work demonstrates the proof-of-principle that the global ITMM problem can be solved with a Krylov solver with similar scalability to that of the stationary solvers.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to Randal Baker and Jon Dahl of Los Alamos National Laboratory for performing calculations on the Redtail computer system, giving us massively parallel results compared to the PARTISN code. Further, the authors thank Tom Evans of Oak Ridge National Laboratory for his assistance in getting us access to and user time on the JaguarPF system. This work has been sponsored by a grant from the National Nuclear Security Administration.

REFERENCES

1. R. S. Baker and K. R. Koch, "An S_n Algorithm for the Massively Parallel CM-200 Computer," *Nuclear Science and Engineering*, **128**, pp.312–320 (1998).
2. R. E. Alcouffe, R. S. Baker, J. A. Dahl, S. A. Turner, and R.C. Ward, *PARTISN: A Time-Dependent, Parallel Neutral Particle Transport Code System*, LA-UR-08-07258, Los Alamos National Laboratory (2008).
3. T. M. Evans, A. S. Stafford, R. N. Slaybaugh, and K. T. Clarno, "Denovo: A New Three-Dimensional Parallel Discrete Ordinates Code in SCALE," *Nuclear Technology*, **171**, pp.171–200 (2010).
4. M. Rosa, *Properties of the S_N-Equivalent Integral Transport Operator and the Iterative Acceleration of Neutral Particle Transport Methods*, PhD Thesis, The Pennsylvania State University, available online at http://www.etd.psu.edu (2007).
5. Y. Y. Azmy, "A New Algorithm for Generating Highly Accurate Benchmark Solutions to Transport Test Problems," *Proceedings of the XI ENFIR / IV ENAN Joint Nuclear Conferences*, Poços de Caldas Springs, MG, Brazil, August 18–22, 1997, on CD-ROM (1997).
6. R. J. Zerr and Y. Y. Azmy, "A Parallel Algorithm for Solving the Multidimensional Within-Group Discrete Ordinates Equations with Spatial Domain Decomposition," *Proceedings of PHYSOR 2010*, Pittsburgh, PA, USA, May 9–14, 2010, American Nuclear Society, LaGrange Park, IL, on CD-ROM (2010).
7. M. Yavuz and E. W. Larsen, "Iterative Methods for Solving x-y Geometry S_N Problems on Parallel Architecture Computers," *Nuclear Science and Engineering*, **112**, pp.32–42 (1992).
8. Y. Saad, *Iterative Methods for Sparse Linear Systems*, 1st Ed., available online http://www-users.cs.umn.edu/~saad/books.html (1996).
9. K. Koch, "Roadrunner Platform Overview," presentation available online at http://www.lanl.gov/orgs/hpc/roadrunner/pdfs/Koch%20-%20Roadrunner%20Overview/RR%20Seminar%20-%20System%20Overview.pdf (2008).
10. "National Center for Computational Sciences – Jaguar," http://www.nccs.gov/computing-resources/jaguar (2010).
11. M. L. Adams and E. W. Larsen, "Fast Iterative Methods for Discrete-Ordinates Particle Transport Calculations," *Progress in Nuclear Energy*, **40**, pp.3–159 (2002).