PERFORMANCE OF PARALLEL APPROXIMATE IDEAL RESTRICTION MULTIGRID FOR TRANSPORT APPLICATIONS

A Thesis

by

JOSHUA THOMAS HANOPHY

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee, Dr. Jim Morel
Committee Members, Dr. Jean Ragusa
Dr. Raytcho Lazarov
Head of Department, Dr. John E. Hurtado

March 2019

Major Subject: Nuclear Engineering

Copyright 2019 Joshua T. Hanophy

ABSTRACT

Algebraic multigrid (AMG) methods have been widely used to solve systems arising from the discretization of elliptic partial differential equations. In serial, AMG algorithms scale linearly with problem size. In parallel, communication costs scale logarithmically with the number of processors. Recently, a classical AMG method based on approximate ideal restriction (AIR) was developed for nonsymmetric matrices. AIR has already been shown to be effective for solving the linear systems arising from upwind discontinuous Galerkin (DG) finite element discretization of advection-diffusion problems, including the hyperbolic limit of pure advection. A new parallel version of AIR, pAIR, has been implemented in the *hypre* library.

In this thesis, pAIR is tested for use solving the source iteration equations of the S_N approximation to the transport equation. The performance is investigated with various meshes in two and three dimensions. Detailed profiling of parallel performance is also conducted to identify the most important areas for algorithm improvements. An improvement to the Local Ideal Approximate Restriction algorithm is introduced and discussed. Weak scaling results to 4,096 processors are presented. These results show total solve growing logarithmically with the number of processors used. Importantly, this result is shown on both uniform grids and unstructured grids in three dimensions. The unstructured mesh did not include reentrant cells.

ACKNOWLEDGMENTS

I would like to thank my graduate committee for taking time to provide feedback. I would also like to thank Ben Southworth for the large amount of help while in learning the pAIR algorithm. Finally, I would thank Ruipeng Li for fielding questions about the pAIR algorithm as implemented in *hypre*.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professors Jim Morel and Jean Ragusa of the Department of Nuclear Engineering and Professor Raytcho Lazarov of the Department of Mathematics.

All work conducted for the thesis was completed by the student independently.

Funding Sources

This material is based upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number(s) DE-NA0002376.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Motivation

Algebraic multigrid (AMG) methods have been widely used to solve systems arising from the discretization of elliptic partial differential equations. In serial, AMG algorithms scale linearly with problem size. In parallel, communication costs scale logarithmically with the number of processors [1]. Recently, a classical AMG method based on approximate ideal restriction (AIR) was developed for nonsymmetric matrices. AIR has already been shown to be effective for solving the linear systems arising from upwind discontinuous Galerkin (DG) finite element discretization of advection-diffusion problems, including the hyperbolic limit of pure advection [2][3]. A new parallel version of AIR, pAIR, has been implemented in the *hypre* library.

In this thesis, the performance of pAIR for solving the S_N equations is investigated. pAIR is tested for use with various meshes in two and three dimensions. pAIR scaling results to 4,096 processors are presented. Detailed profiling of parallel performance is also conducted to identify the most important areas for algorithm improvements. Improvements to the pAIR setup algorithm motivated by the performance investigation are discussed.

1.2 $\mathbf{S_N}$ Approximation and Finite Element Treatment of Transport

The monoenergetic steady state transport equation with isotropic cross sections and an isotropic scattering source is shown in Equation 1.1. For simplicity, only the monoenergetic case is considered here; however, the results should be applicable to the problems where the energy dependency is considered. The S_N equations are derived by first defining an angular quadrature rule. The scalar flux is then defined as shown in Equation 1.2. Evaluating Equation 1.1 at the quadrature points results in M coupled equations that are discrete in direction. This is shown in Equation 1.3. Equation 1.3 could be discretized in space and solved directly; however, for any significant number of directions, the resulting linear system would generally be too large to solve.

$$\vec{\Omega} \cdot \nabla \psi \left(\vec{r}, \vec{\Omega}\right) + \sigma_t \psi \left(\vec{r}, \vec{\Omega}\right) = \frac{\sigma_s}{4\pi} \phi \left(\vec{r}\right) + \frac{q}{4\pi} \tag{1.1}$$

$$\begin{aligned} \phi \left(\vec{r}\right) &= \sum_{m=1}^{M} w_m \psi_m \left(\vec{r}\right) \\ \psi_m \left(\vec{r}\right) &= \psi \left(\vec{r}, \vec{\Omega}_m\right) \end{aligned} \tag{1.2}$$

$$\vec{\Omega} \cdot \nabla \psi_m \left(\vec{r}\right) + \sigma_t \psi_m \left(\vec{r}\right) = \frac{\sigma_s}{4\pi} \phi \left(\vec{r}\right) + \frac{q}{4\pi} \tag{1.3}$$

A common technique used to iteratively solve these equations is source iteration. Source iteration involves lagging the scalar flux. This is shown in Equation 1.4 where n is the iteration index. M independent solves are required to update the scalar flux iterate.

$$\vec{\Omega}_m \cdot \nabla \psi_m^{n+1} \left(\vec{r}\right) + \sigma_t \psi_m^{n+1} \left(\vec{r}\right) = \frac{\sigma_s}{4\pi} \phi^n \left(\vec{r}\right) + \frac{q}{4\pi} \tag{1.4}$$

1.2.1 Upwinded Discontinuous Galerkin Method for Transport

The performance of pAIR will investigated for solving the upwind DG discretization of the source iteration equations. The weak formulation for the steady state transport equation is shown in below

$$\begin{aligned} &- \sum_{T \in T_h} \left(u_h^{n+1}, \Omega \cdot \nabla v_h\right) - \sum_{F \in F_h^i} \left\langle {}^{-}u_h^{n+1}, \Omega \cdot [v_h \mathbf{n}] \right\rangle_F + \sum_{T \in T_h} \left(\sigma_t u_h^{n+1}, v_h\right) + \\ &\left\langle u_h^{n+1}, v_h \Omega \cdot \mathbf{n} \right\rangle_{\Gamma+} = - \left\langle g, v_h \Omega \cdot \mathbf{n} \right\rangle_{\Gamma-} + \sum_{T \in T_h} \left(q_{total}^n, v_h\right) \end{aligned} \tag{1.5}$$

where n and n+1 are iteration indices, $(\cdot, \cdot)_T$ are inner products over the volume and $\langle \cdot, \cdot \rangle_T$ are the inner products over the faces. $[v_h \mathbf{n}] = v_h \mathbf{n}^+ - v_h \mathbf{n}^-$ is the jump term where + and - refer to the upwind and downwind faces respectively. q_{total} is the total source term including a volumetric source and scattering source where the scalar flux is lagged as shown in Equation 1.4. g is the boundary source. Γ^- is the inflow boundary and Γ^+ is the outflow boundary.

For simplicity, only isoparametric multilinear quadrilateral and hexahedral elements are investigated in this thesis. AIR has been shown to be effective for higher order elements [2][3] and it is expected that pAIR is similarly effective although that is not investigated in this thesis.

1.3 Solution of the Transport Equation by Sweeping

The performance of pAIR will be compared with a particular parallel sweeping algorithm. With a structured grid, for many spatial discretizations the system arising from the left side of Equation 1.5 is logically block lower triangular. Sweeps are analogous to directly inverting the block lower triangular matrix. Some aspects of any parallel sweeping algorithm are inherently serial; however, specific ordering of cells and spatial partitioning among the processors allows for parallel algorithms. One popular method for such problem decomposition and cell grouping is the KBA method proposed by Koch, Baker, and Alcouffe [4]. Parallel efficiency of such algorithms is improved by solving multiple directions at a time.

If the directions solved simultaneously are all in the same quadrant or octant, then the resulting method is a natural extension of the method for a single direction. If this is not the case, then the situation might exist where for any particular cell, the conditions are met for different directions to be solved at the same time. A scheduling algorithm is required to dictate in what order the directions are solved in this situation. This situation was analyzed in detail and a provably optimal algorithm presented in [5][6].

On an unstructured grid, methods analogous to the KBA method have not been developed [7]. Parallel sweeping on unstructured grids remains an area of active research. To our knowledge, no detailed performance model analogous to that presented in [5][6] has been developed for parallel sweeping on unstructured grids.

1.4 Introduction to Multigrid

The motivation and basic components of a multigrid method are first introduced from a geometric perspective. Algebraic multigrid is then introduced. Finally, the new AMG algorithm, Approximate Ideal Restriction is introduced.

1.4.1 Motivation for Multigrid Methods and Basic Components

Iterative multigrid methods are widely used to solve the linear systems arising from certain types of problems. The primary components of a multigrid method are smoothing or relaxation and coarse-grid correction. The smoothing step involves one or more iterations of some linear solver. A simple parallelizable method such as Jacobi iteration is often used. For simplicity, first consider an operator A arising from the discretization of a model elliptic PDE on a structured grid. It can be shown that typical iterative methods rapidly attenuate geometrically oscillatory error, but reduce geometrically smooth error arbitrarily slowly. Geometrically smooth error can be accurately represented on a coarser grid, that is a grid with fewer grid points. Solving for these smooth error modes on the coarser grid will be more efficient because there are fewer variables and the error modes that are smooth on a fine grid are less smooth on the coarse grid. This motivates coarse-grid correction. A correction that eliminates smooth error can be computed efficiently on a coarser grid and the correction can be interpolated back to the original grid.

For the system $Au = f$, let u_m be any approximate solution. A relaxation step can be written as $\bar{u}^m = Mu^m + s$. The residual, r, is then calculated $r^m = f - A\bar{u}^m$. Note that $Ae = r$. This equation can be solved approximately for the error. Since the error modes not attenuated by the relaxation step can be represented on a coarser grid, this equation will be solved approximately on the coarse grid.

Different methods exist for determining a coarse grid. In the case of a structured grid, a standard method for determining the coarse grid is to eliminate some points from the fine mesh such that the distance between grid points is doubled in all directions. The relationship between the degrees of freedom in the course grid, $\#\Omega_H$, and the fine grid, $\#\Omega_h$, would be given by $\#\Omega_H \approx \frac{1}{2^d}\#\Omega_h$ where d is the dimension of the problem. The residual from the fine mesh is restricted to the coarse mesh $r_H^m = Rr_h^m$ where R is a restriction operator. The error equation is solved on the coarse mesh $e_H^m = \hat{\kappa} r_H^m$ where κ is the coarse grid operator and $\hat{\kappa}$ is the approximate inverse. Note that it is assumed here that the coarse mesh is still too large for κ to be directly inverted, but if this is possible, κ^{-1} may be computed. The fine mesh correction is interpolated from the coarse error $e_h^m = Pe_H^m$, where P is the interpolation operator, and then the approximate solution is updated to the next iterate $u^{m+1} = u^m + e^m$. This process is summarized in Equation 1.6 where the over bar is meant to indicate the approximate solution after one or more relaxation step. Subtracting Au from both sides of Equation 1.6 leads to the error propagation formula $e^{k+1} = \bar{e}^k + P\kappa^{-1}RA\bar{e}^k$.

$$\begin{aligned} u^{m+1} &= \bar{u}^m + P\kappa^{-1}R\left(b - A\bar{u}^m\right) \\ &= \bar{u}^m + P\kappa^{-1}R\left(Au - A\bar{u}^m\right) \\ &= \bar{u}^m + P\kappa^{-1}RA\bar{e}^m \end{aligned} \tag{1.6}$$

The coarse grid will generally still have a large number of variables and the coarse grid equation needs to be solved approximately. The coarse grid equation can be solved using the same multigrid method described in the previous paragraph where the initial coarse mesh becomes the fine mesh. This process is repeated until the coarse mesh has few enough variables that the coarse grid equation can be solved directly.

1.4.2 Algebraic Multgrid

Algebraic multigrid (AMG) methods generally do not require information about the grid. Coarsening is done with information from the operator itself. There is not necessarily any geometric basis for the coarsening step the in AMG algorithm. For example, algebraically smooth error is simply whatever error modes are not well attenuated in the relaxation step. These error modes may be geometrically smooth for some problems, but this is not the case in general. The terminology from geometric multigrid is used in general to describe similar features of AMG and so the initial matrix is referred to as the fine grid and the coarse grid is a matrix of lower dimension.

An important general difference between geometric multigrid and AMG is that in many geometric multigrid algorithms, the coarsening step is a relatively simple algorithm with direct ties to the problem geometry. With AMG, there is more freedom to define to coarse grid. There is also more freedom in defining the restriction and interpolation operators. A general goal is to define the coarse grid such that it can accurately represent the error modes not well attenuated by the relaxation step on the fine grid. The general goal for constructing the interpolation operator is that it be able to accurately interpolate the coarse grid error correction to the fine grid.

1.4.2.1 Coarsening and Definition of Coarse Grid Operator

Each variable in the linear system is assigned to be either a C-point or a F-point. The top level or fine mesh is the union of C-points and F-points while the coarse level contains only the C-points. The motivation behind such a partitioning is to find variables whose value can be accurately interpolated from the value at other variables. That is, the goal is to find variables which depend on each other and then solve for a subset of these in the coarse grid. The coarse grid correction can then be interpolated to the other variables in the fine grid. A commonly used term to describe dependence is strong connection. Coarsening is based on determining variables that are strongly connected.

A commonly used algorithm for determining strong connections is based on a strength of connection (SOC) parameter. This is shown below where ϵ_{str} is the SOC parameter.

$$-a_{ij} \geq \epsilon_{str} max_{i \neq j} \left(|a_{ij}| \right) \tag{1.7}$$

With reference to Equation 1.7, variable i is strongly connected to variable j if the coefficient a_{ij} is greater than or equal to the maximum off diagonal coefficient in the row times the SOC parameter value.

The Galerkin coarse grid operator is commonly used in AMG. This operator is defined as the matrix product RAP. Many AMG methods are based on approximating an ideal interpolation operator P_{ideal}. Ideally, the error after relaxation on the fine mesh would be in the range of the interpolation operator. Then the coarse grid correction can be interpolated exactly. The ideal interpolation operator will generally not be sparse and is difficult to compute. Many AMG algorithms are focused on computing an approximation to P_{ideal}. The restriction operator is then defined as $R = P^T$ so that when A is symmetric, the coarse grid operator RAP is also symmetric. When A is not symmetric, using $R = P^T$ is less meaningful and not used in the AIR algorithm. Construction of R and P in the AIR algorithm is described in subsequent sections.

1.4.3 Introduction to Approximate Ideal Restriction (AIR) multigrid

It is assumed that A can be permuted as shown below where the F-points are ordered before the C-points. The A_{FF} and A_{CC} block then indicate connections between F-points and C-points respectively.

$$\begin{bmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{bmatrix} \tag{1.8}$$

The restriction operator R and the interpolation operator P can then be written as

$$P = \begin{bmatrix} Z \\ I \end{bmatrix} \tag{1.9}$$

$$R = \begin{bmatrix} W & I \end{bmatrix} \tag{1.10}$$

Analogous to there being an ideal interpolation operator, there is an ideal restriction operator whose purpose is to eliminate error modes . In Equation 1.11, the error at F-points e_F has been written as $Ze_C + \delta e_F$ where δe_F is error not in the range of interpolation.

$$e = \begin{bmatrix} e_F \\ e_C \end{bmatrix} = \begin{bmatrix} Ze_C + \delta e_F \\ e_C \end{bmatrix} \tag{1.11}$$

The goal of ideal R is to eliminate δe_F. Equation 1.11 can be placed into the error propagation formula. This is shown below:

$$\begin{aligned} e^{m+1} &= \begin{pmatrix} e_F^m \\ e_C^m \end{pmatrix} - P\left(RAP\right)^{-1} RA \left[\begin{pmatrix} Ze_C^m \\ e_C^m \end{pmatrix} + \begin{pmatrix} \delta e_F^m \\ 0 \end{pmatrix} \right] \\ &= \begin{pmatrix} e_F^m \\ e_C^m \end{pmatrix} - P\left(RAP\right)^{-1} RA \left[\begin{pmatrix} Pe_c^m \end{pmatrix} + \begin{pmatrix} \delta e_F^m \\ 0 \end{pmatrix} \right] \\ &= \begin{pmatrix} e_F^m \\ e_C^m \end{pmatrix} - Pe_C^m - P\left(RAP\right)^{-1} RA \begin{pmatrix} \delta e_F^m \\ 0 \end{pmatrix} \end{aligned} \tag{1.12}$$

Therefore, the condition to eliminate δe_F is that

$$RA \begin{pmatrix} \delta e_F^m \\ 0 \end{pmatrix} = 0 \tag{1.13}$$

If this equality is satisfied, then it will be possible to accurately calculate the error at C-points on the coarse grid. In the AIR algorithm, the accurate error correction at the C-points is transferred back to the fine grid via simple point-wise interpolation. This updates the C-points in the fine grid, but not the F-points. The relaxation step is then applied several times over only the F-points on the fine grid. In this way, the AIR algorithm discussed here is reduction based. A subset of the original problem is selected that can be solved accurately.

1.4.4 Local Approximate Ideal Restriction (LAIR)

This section includes a more detailed discussion on constructing the Local Approximate Ideal Restriction (LAIR) operator. The goal is to approximately satisfy the equality shown in Equation 1.13. For each C-point, a local neighborhood of F-points is selected. Let the neighborhood of nearby F-points for the i-th C-point be denoted $\mathcal{R}_i$. With reference to the system shown below, the

F-points in $\mathcal{R}_i$ are labeled j_n.

$$\begin{bmatrix} a_{11} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & a_{1N} \\ \vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\ \vdots & \dots & a_{j_1j_1} & a_{j_1j_2} & a_{j_1i} & a_{j_1j_3} & a_{j_1j_4} & \dots & \vdots \\ \vdots & \dots & a_{j_2j_1} & a_{j_2j_2} & a_{j_2i} & a_{j_2j_3} & a_{j_2j_4} & \dots & \vdots \\ \vdots & \dots & a_{ij_1} & a_{ij_2} & a_{ii} & a_{ij_3} & a_{ij_4} & \dots & \vdots \\ \vdots & \dots & a_{j_3j_1} & a_{j_3j_2} & a_{j_3i} & a_{j_3j_3} & a_{j_3j_4} & \dots & \vdots \\ \vdots & \dots & a_{j_4j_1} & a_{j_4j_2} & a_{j_4i} & a_{j_4j_3} & a_{j_4j_4} & \dots & \vdots \\ \vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\ a_{N1} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & a_{NN} \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ F_{j_1} \\ F_{j_2} \\ C_i \\ F_{j_3} \\ F_{j_4} \\ \vdots \\ \vdots \end{bmatrix} \tag{1.14}$$

If error not in the range of interpolation at any F-point ij is written as δ_{ij}, note then that the residual becomes

$$\begin{bmatrix} \vdots \\ e_i a_{j_1i} + \delta_{j_1} a_{j_1j_1} + \delta_{j_2} a_{j_1j_2} + \delta_{j_3} a_{j_1j_3} + \delta_{j_4} a_{j_1j_4} \\ e_i a_{j_2i} + \delta_{j_1} a_{j_2j_1} + \delta_{j_2} a_{j_2j_2} + \delta_{j_3} a_{j_2j_3} + \delta_{j_4} a_{j_2j_4} \\ e_i a_{ii} + \delta_{j_1} a_{ij_1} + \delta_{j_2} a_{ij_2} + \delta_{j_3} a_{ij_3} + \delta_{j_4} a_{ij_4} \\ e_i a_{j_3i} + \delta_{j_1} a_{j_3j_1} + \delta_{j_2} a_{j_3j_2} + \delta_{j_3} a_{j_3j_3} + \delta_{j_4} a_{j_3j_4} \\ e_i a_{j_4i} + \delta_{j_1} a_{j_4j_1} + \delta_{j_2} a_{j_4j_2} + \delta_{j_3} a_{j_4j_3} + \delta_{j_4} a_{j_4j_4} \\ \vdots \end{bmatrix} \tag{1.15}$$

It is clear then, that in reference to these points, that a unit change in error at point F_{j_n} will change the residual at point C_i by a_{ij_n}. Note that R applied to the residual, Rr, is seen to be

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & w_{ij_1} & w_{ij_2} & 1 & w_{ij_3} & w_{ij_4} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ e_i a_{j_1 i} + \delta_{j_1} a_{j_1 j_1} + \delta_{j_2} a_{j_1 j_2} + \delta_{j_3} a_{j_1 j_3} + \delta_{j_4} a_{j_1 j_4} \\ e_i a_{j_2 i} + \delta_{j_1} a_{j_2 j_1} + \delta_{j_2} a_{j_2 j_2} + \delta_{j_3} a_{j_2 j_3} + \delta_{j_4} a_{j_2 j_4} \\ e_i a_{ii} + \delta_{j_1} a_{ij_1} + \delta_{j_2} a_{ij_2} + \delta_{j_3} a_{ij_3} + \delta_{j_4} a_{ij_4} \\ e_i a_{j_3 i} + \delta_{j_1} a_{j_3 j_1} + \delta_{j_2} a_{j_3 j_2} + \delta_{j_3} a_{j_3 j_3} + \delta_{j_4} a_{j_3 j_4} \\ e_i a_{j_4 i} + \delta_{j_1} a_{j_4 j_1} + \delta_{j_2} a_{j_4 j_2} + \delta_{j_3} a_{j_4 j_3} + \delta_{j_4} a_{j_4 j_4} \\ \vdots \end{bmatrix}$$

where w_{ij} is the restriction weight at point ij of R. The value of the weights is determined by the goal of canceling the contributions of each δ_{ij} from the residual. This leads to the linear system shown below.

$$a_{ij} + \sum_{k \in \mathcal{R}_i} w_{ik} a_{kj} = 0, \quad \forall j \in \mathcal{R}_i \tag{1.16}$$

The size of $\mathcal{R}_i$ determines the quality of the approximate R to the ideal R. As more F-points are considered, the approximation quality increases, but the cost to compute R also increases. For distance-1 LAIR, the F-points considered for the i^{th} C-point are those that are in the same row. Distance-2 LAIR considers the next level of F-F connections. Notice that as written in Equation 1.15, for row j_1, no coefficients were considered to the left of coefficient $a_{j_1 j_1}$, but there would generally be coefficients not pictured. Distance-2 LAIR accounts for these additional coefficients when building R. Note that not accounted for in distance-1 LAIR is the error contribution from F-points which are not in the row of the i-th C-point, but that are in a row for any F-point in $\mathcal{R}_i$, that is in any row j_n. Construction of the distance-2 approximate ideal restriction operator eliminates the error contribution from these points as well. For example, let $\mathcal{R}_{j_n}$ be the neighborhood of F-points connected to point $\mathcal{R}_{j_i}$. Then restriction weights w_{ik} are added for each $\{k \mid k \in \mathcal{R}_{j_n}, k \notin \mathcal{R}_i\}$ where $\mathcal{R}_i$ are the distance-1 points.

Instead of selecting the neighborhood of F-points from the matrix A, the neighborhood is selected from a SOC matrix calculated for A. This increases the sparsity of R. A neighbor is considered if it is strongly connected to the C-point. The strength of connection test used when building R is similar to the classical coarsening SOC test. That is

$$|a_{ij}| \geq \epsilon_R max_{i \neq j} \left(|a_{ij}| \right) \tag{1.17}$$

where a new SOC parameter is introduced, ϵ_R, that will not generally be the same as the parameter used for coarsening. Note also that the absolute value of coefficients is used instead of the negative value.

The general procedure for calculating R is then

1. An SOC matrix is created for A. The strength of connection parameter, or more generally, the algorithm used can be different from that used for the original C/F splitting.

2. The SOC matrix can be used to establish the sparsity pattern for R. Next, the indices of strong neighbors for each C-point and, for distance-2 AIR, the neighborhood for each strong neighbor, is determined from the SOC matrix.

3. For each row of R, the system shown in Eq. 1.16 by taking the coefficients from A. The system is solved either directly or iteratively for the weights.

2. METHODOLOGIES

2.1 deal.II Model

The transport program constructed to test pAIR was created primarily using the deal.II library. deal.II is a fully functional finite element library [8]. It includes capabilities to generate a mesh or read from a mesh data file as well as construct and solve a finite element system. For distributed data structures and distributed linear algebra, deal.II relies on separate libraries. For domain decomposition and storage of a fully distributed mesh, deal.II relies on the p4est library [9]. For fully distributed matrices and vectors, deal.II can use petsc or Trilinos. The Trilinos library was used in the program constructed as part of this thesis [10]. Trilinos can interface with *hypre* in several ways. For this program, the ifpack package was used [11]. The transport code created uses source iteration to solve the S_N equations. Shown in Figure 2.1 is an example scalar flux result.

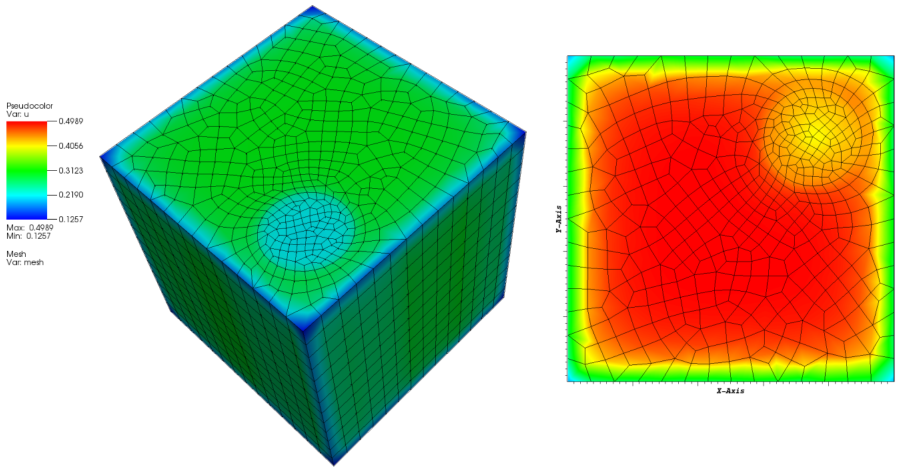


Figure 2.1: Example fully converged scalar flux result for an unstructured grid from the test program created with deal.II

2.2 Generation of Performance Data

The *hypre* library is already able to report some useful metrics about the AMG hierarchy created by the pAIR algorithm. It reports the total time taken, grid and operator complexities, and measured convergence factors. It can also report the amount of time taken coarsening, building R, building P, and building RAP. For more detailed performance information, the profiling library TAU has been used [12]. Of primary interest is the relative costs of communication versus local computations, as well as the parts of the algorithm responsible for bottlenecks in performance.

2.3 Verification with MMS

The correctness of the transport program created to test pAIR was verified using the method of manufactured solution (MMS). An angular flux is assumed which can be separated into an isotropic function and a function that is linearly anisotropic that integrates to one. In 3 dimensions $\psi(x, y, z, \vec{\Omega}) = \Phi(x, y, z)\Psi(x, y, z, \vec{\Omega})$. The assumed scalar flux is $\Phi(x, y, z)$. $\Phi(x, y, z)$ is shown in Eq. 2.1 and $\Psi(x, y, z, \vec{\Omega})$ is shown in Eq. 2.2 where μ, η, and ζ are the direction cosines. This assumed forms for the angular flux and scalar flux are placed in the transport equation and this equation is solved for the volumetric source term. The result of this is shown in Eq. 2.3.

$$\Phi = \sin\left(\frac{x\pi}{L_x}\right)\sin\left(\frac{y\pi}{L_y}\right)\sin\left(\frac{z\pi}{L_z}\right) \tag{2.1}$$

$$\Psi(x, y, \vec{\Omega}) = \left(1 + 2\mu\left(x - \frac{L_x}{2}\right)\right)\left(1 + 2\eta\left(y - \frac{L_y}{2}\right)\right)\left(1 + 2\zeta\left(z - \frac{L_z}{2}\right)\right) \tag{2.2}$$

$$q = \langle \mu, \eta, \zeta \rangle \cdot \langle \frac{\partial(\Phi\Psi)}{\partial x}, \frac{\partial(\Phi\Psi)}{\partial y}, \frac{\partial(\Phi\Psi)}{\partial z} \rangle - \frac{\sigma_s}{4\pi}\Phi + \sigma_t\Phi\Psi \tag{2.3}$$

In two-dimensions, the S_2 Chebyshev quadrature set with 4 directions was used. In three dimensions, the S_4 square Chebyshev-Legendre quadrature set was used [13]. The results for two-dimensions are shown in Figure 2.2 and for three-dimensions in Figure 2.3.

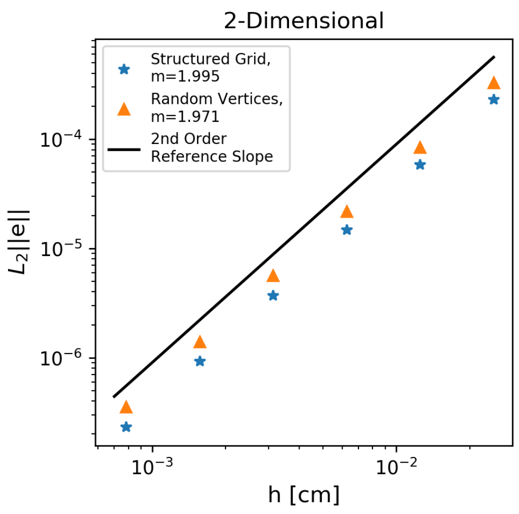


Figure 2.2: Numerical results showing the L2 norm of the scalar flux error for the MMS test of a 2-dimensional problem

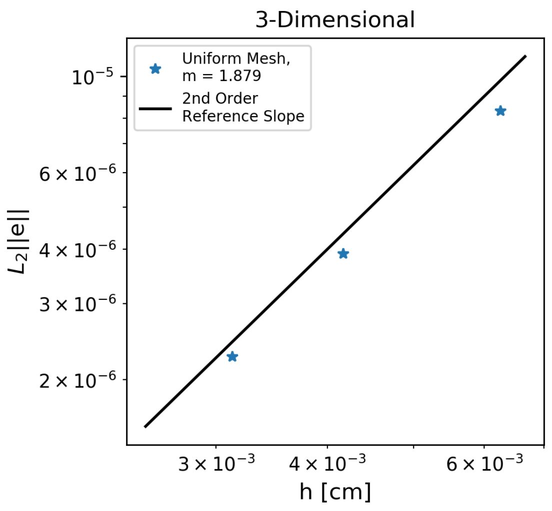
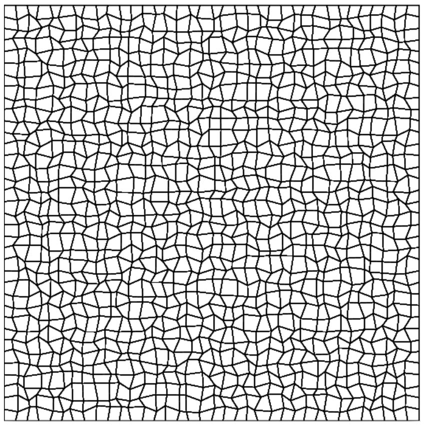


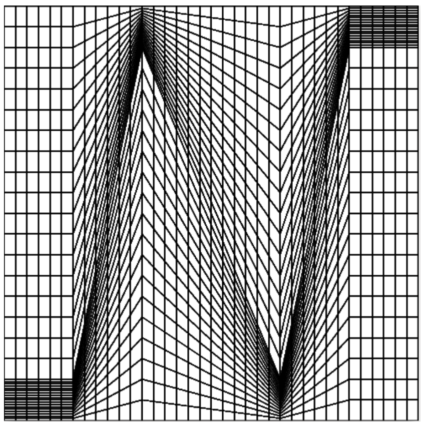
Figure 2.3: Numerical results showing the L2 norm of the scalar flux error for the MMS test of a 3-dimensional problem with a structured grid

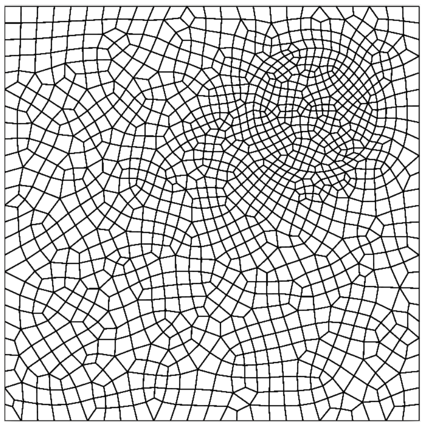
2.4 Meshes

In two dimensions, the performance of the pAIR solver is tested on four different meshes. The first is a uniform structured mesh. The other three meshes are shown in a Figure 2.4. The unstructured mesh was generated by meshing the surface structure shown in Figure 2.5 using the

Gmsh mesh generator program. The zmesh tested in two-dimensions is logically structured and meant primarily to test high aspect ratio elements [14]. In three dimensions, two different mesh types are investigated. These are a uniform structured grid and an unstructured extruded mesh. The unstructured grid is shown in Figure 2.4.


(a) randomized vertices


(b) zmesh [14]


(c) unstructured

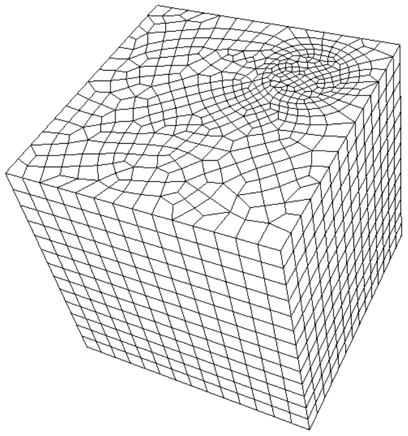

(d) extruded unstructured

Figure 2.4: Meshes tested

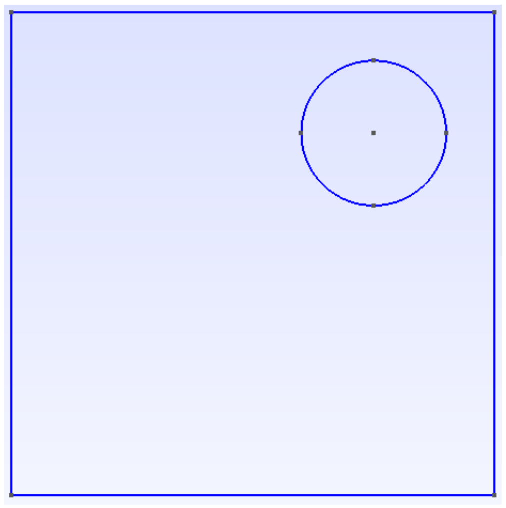


Figure 2.5: Surface configuration used to create the unstructured mesh

3. pAIR ON DIFFERENT GRIDS AND DOMAINS

3.1 Two Dimensions

3.1.1 Test Description

An initial assessment of the importance of material composition to the pAIR algorithm was performed. Table 3.1 lists the different material compositions tested. Figure 3.1 depicts the box and banded configurations. The square domain tested had a side length equal to 1 cm and a volumetric source of 1 $\frac{n}{cm^2}$. The inhomogeneous cross sections were implemented such that the cross section is constant within each cell. For nonuniform meshes, this was accomplished, for each cell, by checking whether any part of the cell was in a banded region or the inner box region and if so, setting the cross section within the entire cell to the appropriate value.

A single node of the computational cluster containing 36 processors was used for the tests presented in this section. The number of processors used was arbitrary. For initial testing, it was desired to use a small number of processors while still exercising the parallel components of the new pAIR implementation. The meshes shown in Figure 2.4 were used as the initial global coarse mesh. These meshes were partitioned across the 36 processors and the meshes were then locally uniformly refined until there were approximately 200,000 degrees of freedom per processor. The pAIR solver parameters used for the tests are presented in Table 3.2.

Table 3.1: Different material configurations tested both structured grids and a grid with randomized vertices

Material Type	Absorption Cross Section
Homogeneous	0.01 cm^{-1}
	1.0 cm^{-1}
	100.0 cm^{-1}
Box1	Inner Box 0.01 cm^{-1}, Outer Box 100.0 cm^{-1}
Box2	Inner Box 100.0 cm^{-1}, Outer Box 0.01 cm^{-1}
Banded	Region 1 0.01 cm^{-1}, Region 2 100.0 cm^{-1}

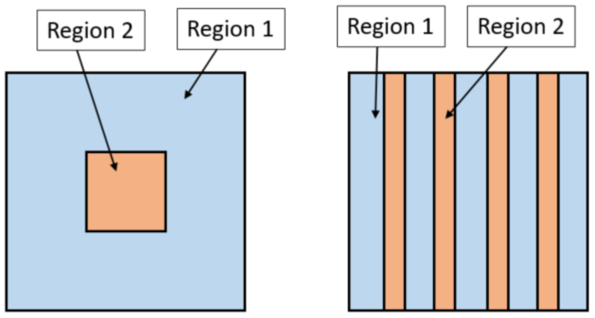


Figure 3.1: Schematic of the Box1, Box2, and Banded configuration

Table 3.2: pAIR solver options and parameters used for the two-dimensional tests

Item	**Value**
Relaxation	Pointwise Jacobi
Prerelax	1-iteration over all points
Postrelax	3-iterations over F-points
LAIR Method	Distance-2
LAIR SOC Parameter ϵ_R	5.0e-3
Filtering Threshold	1.0e-4
Coarsening	Falgout coarsening
Coarsening SOC Parameter ϵ_{str}	5.0e-3

3.1.2 Convergence Factors and Solve Time

Figure 3.2 shows convergence factors for the structured grid and grid with randomized vertices. The same result is shown for the zmesh grid and unstructured mesh in Figure 3.3. The convergence factors for the unstructured mesh are similar to those of the two logically rectangular grids.

For all mesh types investigated, the box1 configuration is very similar to the homogeneous domain test with a cross section of 0.01 cm^{-1}. With reference to Table 3.1, the box1 configuration has a large outer transparent region and a smaller inner opaque region. An important factor for the solver appears to be amount of optically thin cells. Therefore, conclusions based on testing homogeneous domains with different absorption cross sections should be generally applicable.

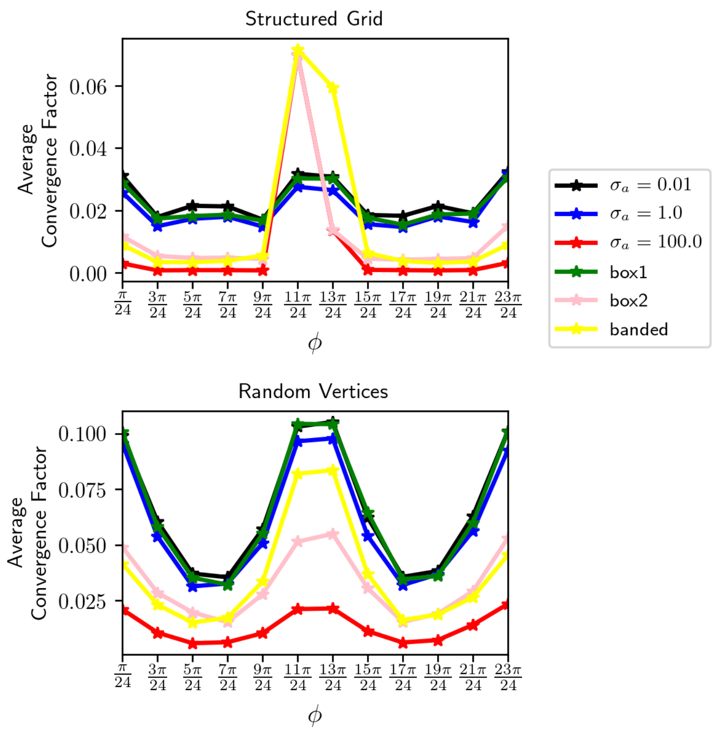


Figure 3.2: Convergence factors for both a structured grid and grid with random vertices shown for each angular flux direction solved

Figure 3.4 shows timings for the pAIR setup phase as well at the time spent in the solving phase for a structured grid and grid with randomized vertices. The same result is shown for the zmsh grid and grid with random vertices in Figure 3.5. Again, the results for the box1 configuration and the optically thin homogeneous domain are very similar. In all cases, the setup phase is taking more time than the solution phase. The time it takes to solve the system is a function of the convergence factor, desired convergence tolerance, as well as characteristics of the multigrid hierarchy. The convergence factors were generally poorer for transparent problems and so the increase in total solve time is expected. The setup time is also generally greater for transparent problems.

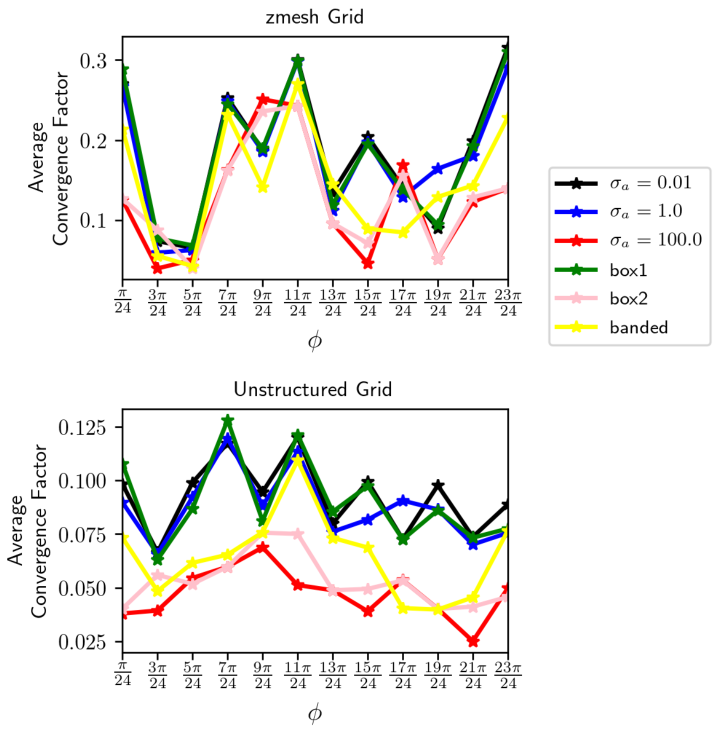


Figure 3.3: Convergence factors for both the zmesh and unstructured mesh shown for each angular flux direction solved

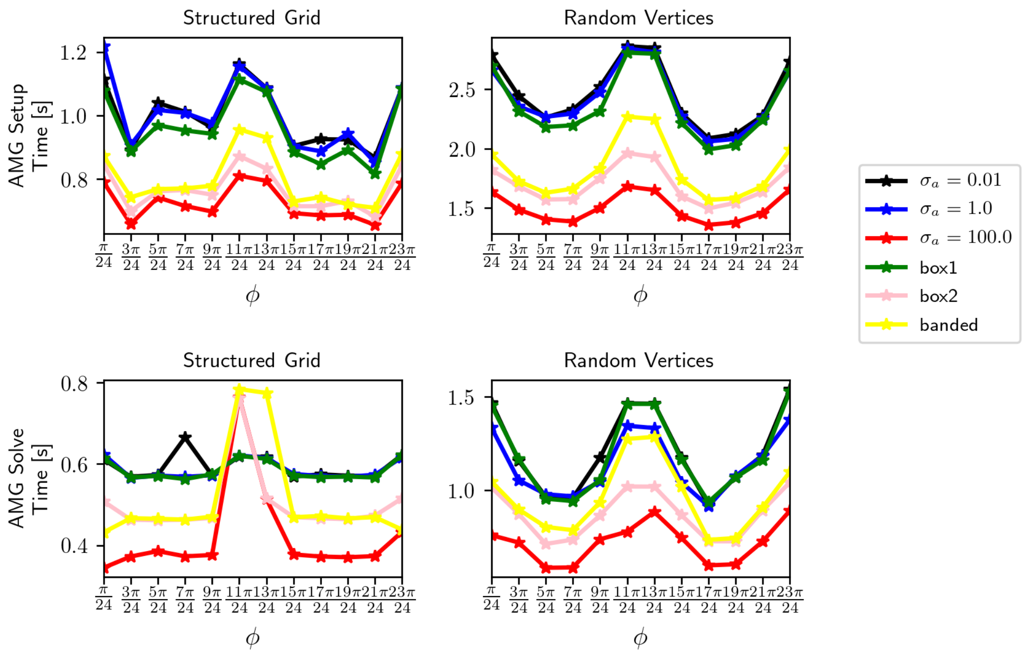


Figure 3.4: AMG setup time and cycle time for both a structured grid and grid with random vertices shown for each angular flux direction solved

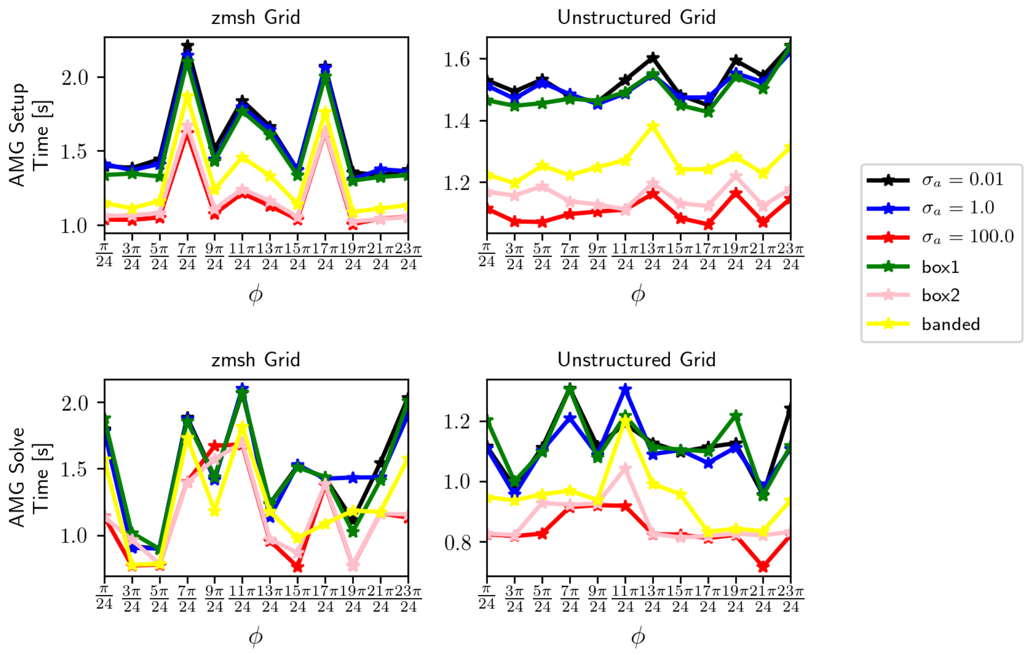


Figure 3.5: AMG setup time and cycle time for both the zmesh and unstructured mesh shown for each angular flux direction solved

3.2 Three Dimensions

3.2.1 Test Description

In three dimensions two mesh types were tested. These are a uniform mesh and the unstructured mesh shown in Figure 2.4. For the uniform mesh, the box and banded configurations shown in Figure 3.1 were extended to three-dimensions. For the box configuration, this was accomplished by turning the inner box into a cube surrounded surrounded by a larger cube. For the banded configuration, the two-dimensional configuration shown was extruded into the third dimension. For the unstructured mesh a Cylinder1 and Cylinder2 configuration were tested. Table 3.3 shows the cross section and source configuration for these tests. The pAIR solver options and parameters used for the tests presented in this section are shown in Table 3.4.

Table 3.3: Material and source configurations tested with the unstructured three dimensional mesh. With reference to Figure 2.5, the cylinder region is the region inside the circle or cylinder when extruded and the outer region is everything else.

Material Type	**Absorption Cross Section**	**Volumetric Source**
Cylinder1	cylinder 0.01 cm^{-1} outer region 100.0 cm^{-1}	cylinder 0.0 $\frac{n}{cm^3 s}$ outer region 1.0 $\frac{n}{cm^3 s}$
Cylinder2	cylinder 100.0 cm^{-1} outer region 0.01 cm^{-1}	cylinder 1.0 $\frac{n}{cm^3 s}$ outer region 0.0 $\frac{n}{cm^3 s}$

Table 3.4: pAIR solver options and parameters used for the three-dimensional tests with a uniform grid

Item	**Value**
Relaxation	Pointwise Jacobi
Prerelax	1-iteration over all points
Postrelax	3-iterations over F-points
LAIR Method	Distance-2
LAIR SOC Parameter ϵ_R	0.1
Filtering Threshold	1.0e-4
Coarsening	Falgout coarsening
Coarsening SOC Parameter ϵ_{str}	0.25

3.2.2 Convergence Factors and Solve Time

Convergence factors for the configurations tested are shown in Figure 3.6. The total time spent in pAIR setup phase as well as the solution phase for each test is shown in Figure 3.7. Similarly to the two-dimensional tests, for both the three-dimensional uniform grid and unstructured grid, convergence factors and total time taken are principally affected by the total amount of optically thin cells.

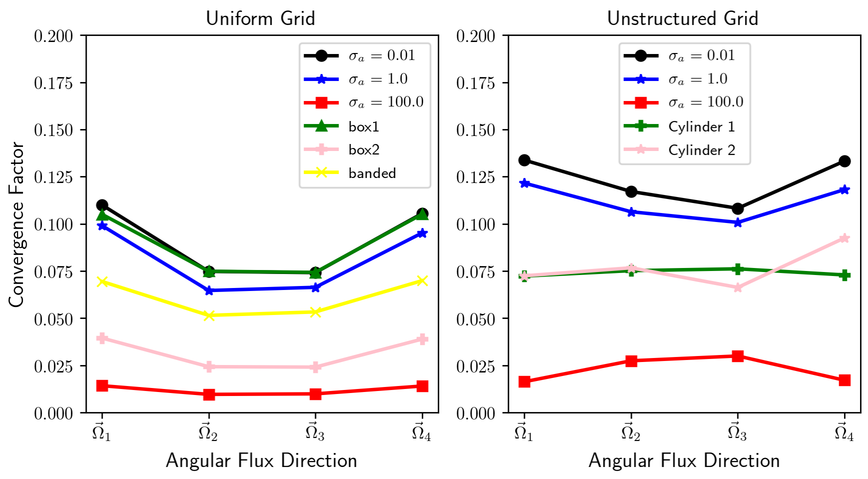


Figure 3.6: Convergence factors for both the uniform mesh and the unstructured mesh shown for four different directions

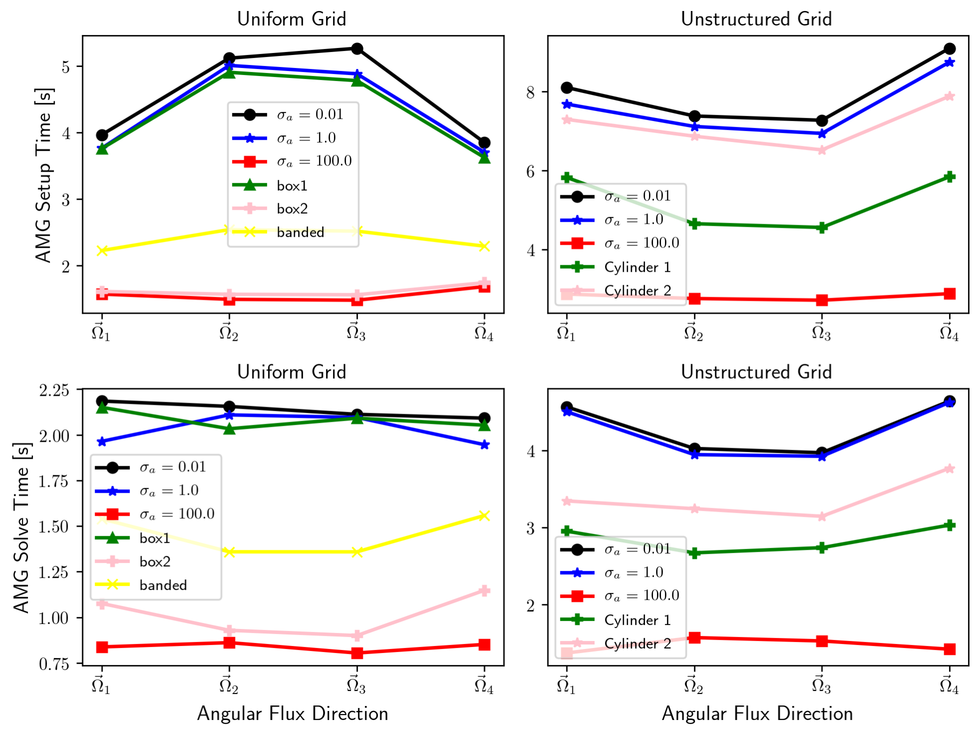


Figure 3.7: Time spent in the setup phase and solution phase for both the uniform mesh and the unstructured mesh shown for four different directions

3.3 Conclusion

A principal factor in pAIR performance for the grids tested appears to be the amount of optically thin cells. Therefore, testing homogenous domains with different absorption cross sections should provide useful results. Only the simple pointwise Jacobi smother was used in these tests. pAIR was able to reliably solve the transport equation on a variety of meshes.

4. PARAMETERS STUDY AND PERFORMANCE CHARACTERISTICS

The performance of an AMG method is generally dependent on the values selected for the various parameters utilized. Additionally, pAIR, like other AMG methods, depends on distinct algorithms that accomplish tasks in the overall method, and different algorithms can be used to accomplish the same task. For example, using different coarsening algorithms can have a dramatic effect on the parallel scalability of an AMG algorithm [1]. Although the full space of possible parameter values and algorithm choices will not be explored, efforts are made to pick the best parameter values and algorithms for the problems being solved. This section is not an exhaustive study of the parameter space, but provides a basis for parameter selections made for the scaling tests and also provides some initial performance results in connection with the parameter selections.

4.1 SOC Parameter for Building R

Construction of R using distance-1 and distance-2 LAIR is discussed in Section 1.4.4. In general, convergence factors will improve as the SOC parameter ϵ_R is made smaller. However, the time it takes to build and use R will generally increase as more connections are considered. This affect in apparent in two dimensions as well as three dimensions, although the increase in cost as ϵ_R is decreased is less significant in two-dimensions since there are fewer connections in general. Presented in this section are results that demonstrate the effect of ϵ_R on convergence factors and overall time for problems relevant to the scaling tests presented in Section 6.

Figure 4.1 and Figure 4.2 show the total pAIR time and setup time for different ϵ_R parameter values for distance-2 LAIR and distance-1 LAIR respectively. Note that each plot shows four different discrete directions to illustrate that trends are similar for any particular direction. The increase in total time for pAIR as ϵ_R decreases is more drastic for distance-2 LAIR. As shown in Figure 4.3, the convergence factors are generally better for distance-2 LAIR as expected. Clearly, for the scaling tests presented Section 6, the largest value of ϵ_R that still provides for reasonable convergence should be used.

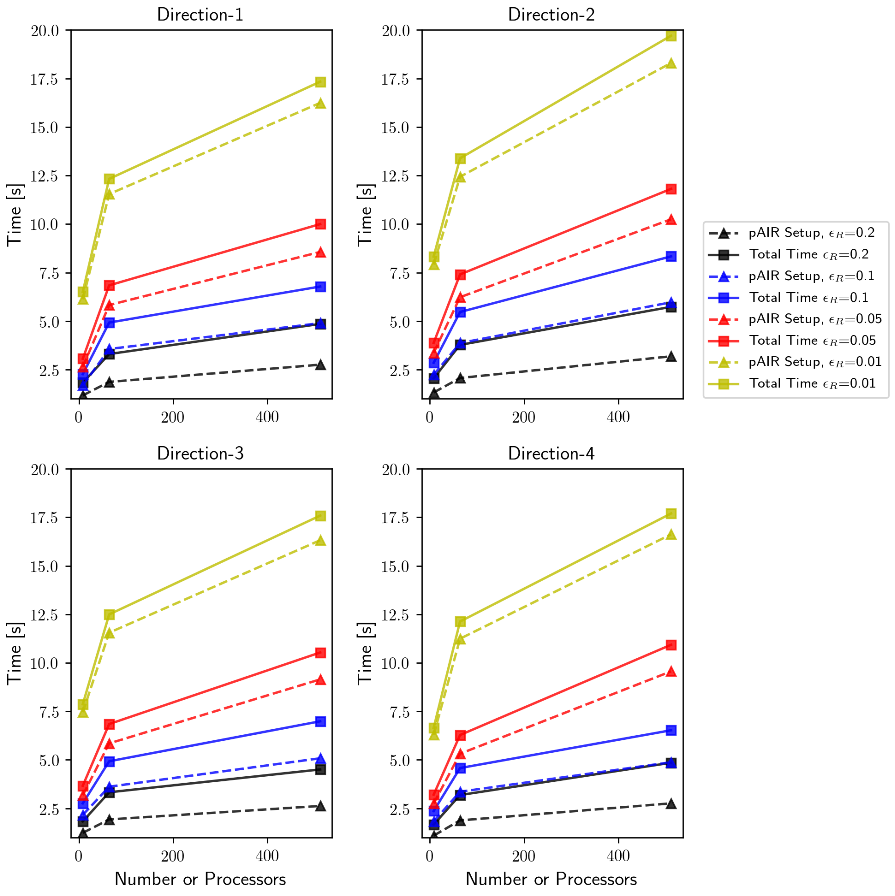


Figure 4.1: Time for single direction solves with different ϵ_R values and distance-2 LAIR. Results were generated from an unstructured three-dimensional grid.

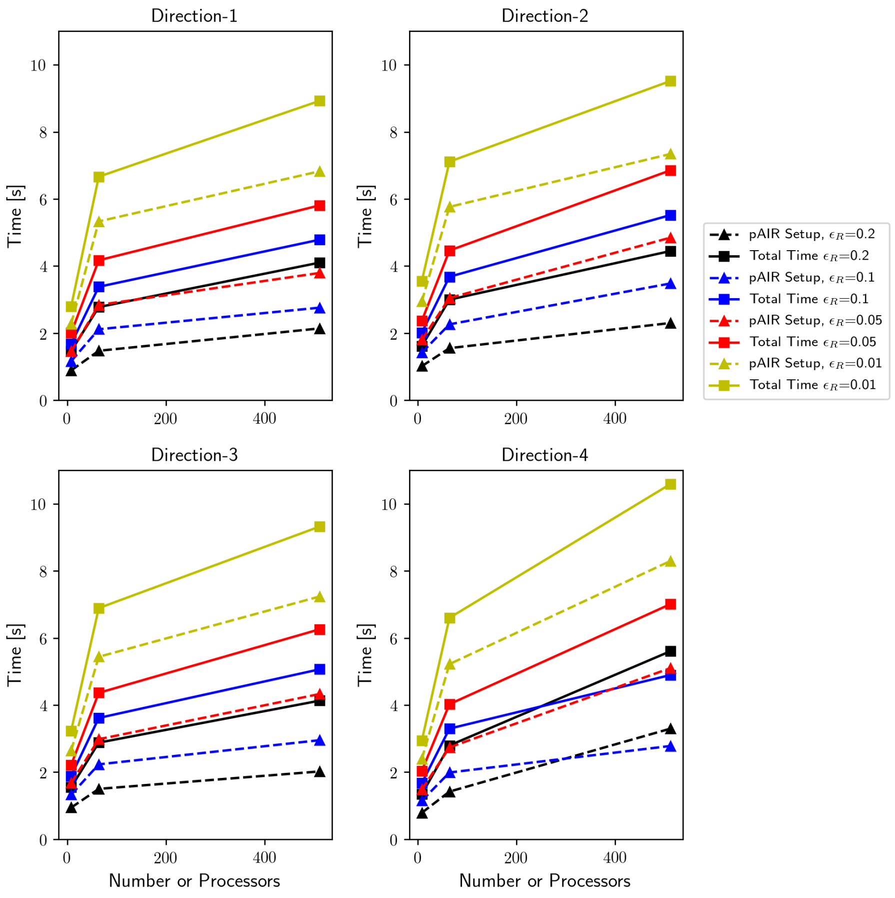


Figure 4.2: Time for single direction solves with different ϵ_R values and distance-1 LAIR. Results were generated from an unstructured three-dimensional grid.

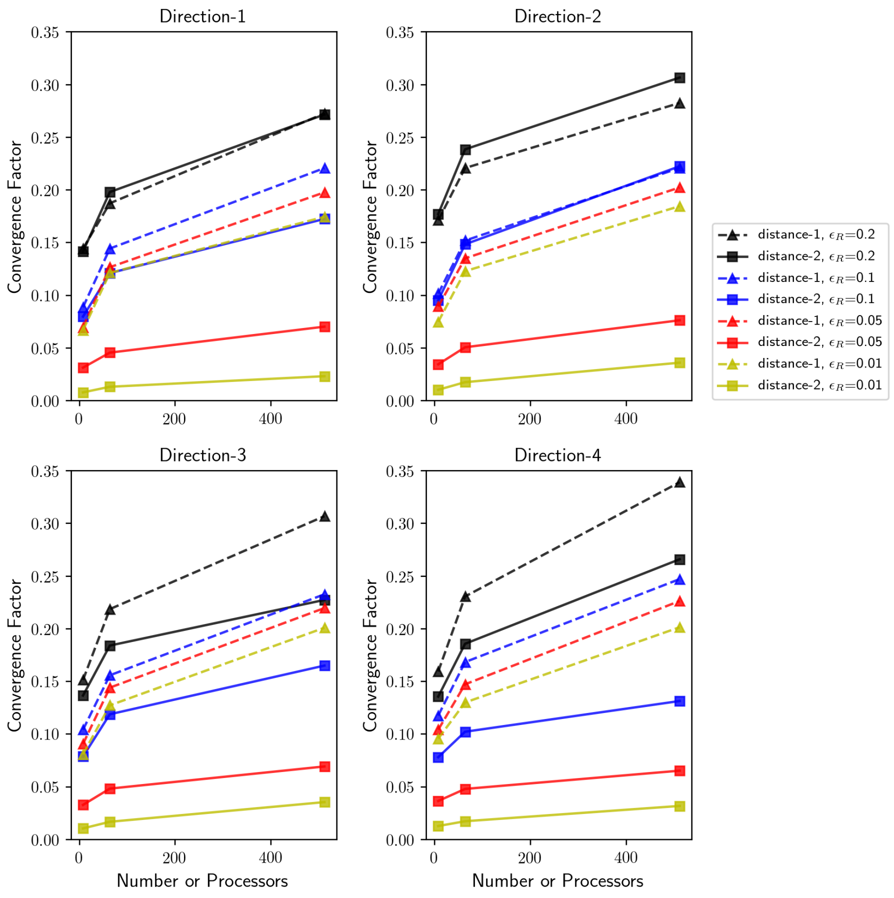


Figure 4.3: Convergence factors for single direction solves with different ϵ_{str} values. Distance-1 LAIR is compared with distance-2 LAIR. Results were generated from an unstructured three-dimensional grid.

4.2 Coarsening SOC Parameter and Building of R

Results related to the coarsening SOC parameter, ϵ_{str}, are presented in this section. The SOC parameter is used in the classical coarsening algorithm to determine strong connections between variables. This is discussed in Section 1.4.2.1. The parameter is investigated in this section for two reasons. First, results are presented to provide some basis to select the best parameter value for larger scaling tests. Secondly, the sensitivity of performance results to the SOC parameter is of general interest. There a multiple algorithms to determine a C/F splitting of the grid based on the strength of connection test, and *hypre* allows the user to select from several different options. For the results presented in this section, Falgout coarsening was used [15, 16]. It is possible that these the trends presented related to the value of ϵ_{str} would change significantly if different coarsening algorithms are used. For example, Figure 4.12 shows that for all other parameters held constant, changing the coarsening algorithm can significantly impact growth in the stencil size.

4.2.1 2 Dimensions

Results for distance-2 LAIR are presented in this section. Figure 4.4 shows average convergence factors and total AMG time for a single direction solve in two dimensions with a uniform grid. Results are shown for a void and a strong absorber. Figure 4.5 shows the same results from a grid with randomized vertices. Results from the uniform grid show that convergence factors improve for very small ϵ_{str} values. However, the convergence factor is small in all cases and the improvement is not significant. However, Figure 4.4 shows that the total AMG time for solution is significantly lower for the smaller ϵ_{str} values. In contrast to the uniform grid result, the result for the perturbed mesh shows that the convergence factor is much worse for small ϵ_{str} values. Despite the poorer convergence factors, the overall solve time is still lower when a small ϵ_{str} value is used.

Figure 4.6 and Figure 4.7 show the time spent coarsening and constructing R for the uniform grid and perturbed grid respectively. The trends shown in these two plots are similar. Note that results are shown for both a vacuum and strong absorber. As the value of ϵ_{str} is decreased, the coarsening time increases. However, there is a much greater decrease in the time to build the restriction operator.

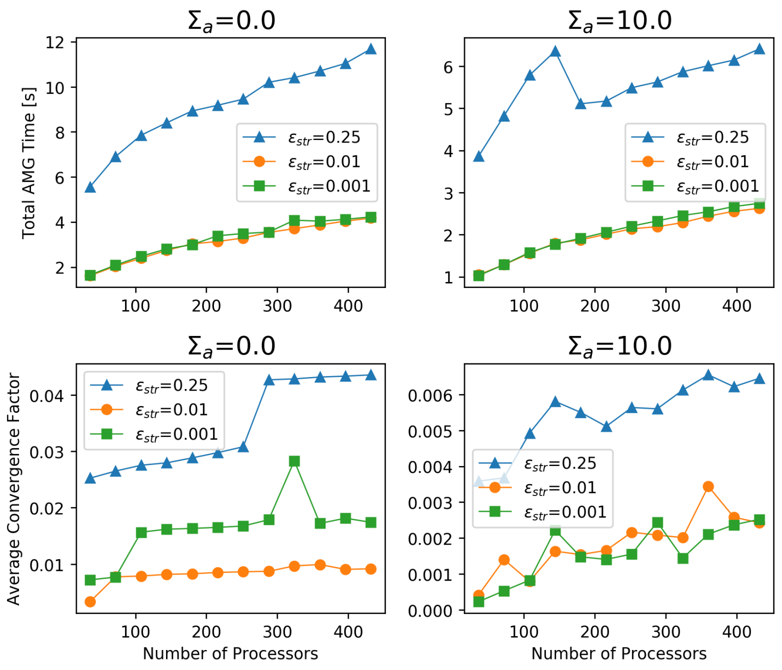


Figure 4.4: Average convergence factors and total AMG time shown for different ϵ_{str} values with a uniform grid. Results are shown for two different cross sections.

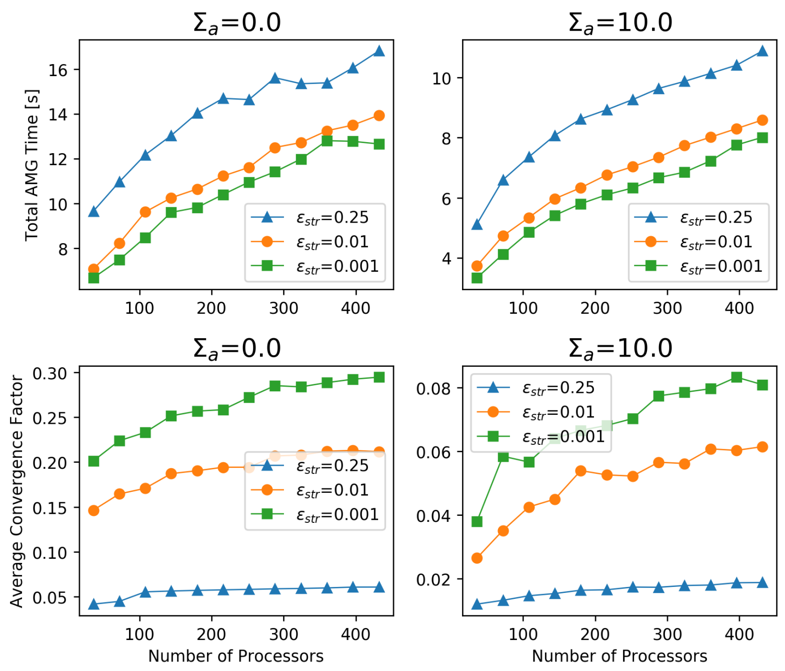


Figure 4.5: Average convergence factors and total AMG time shown for different ϵ_{str} values for a grid with random vertices. Results are shown for two different cross sections.

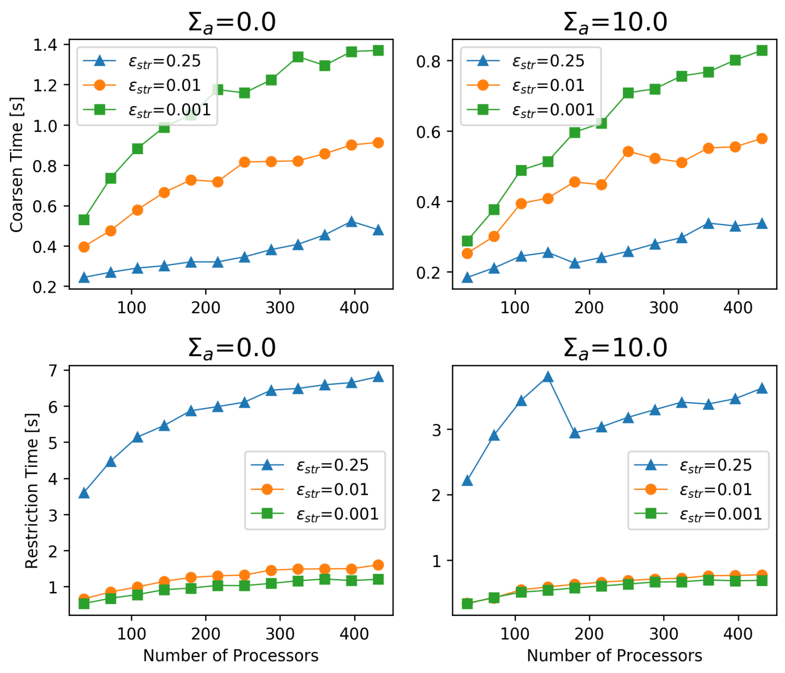


Figure 4.6: Time for coarsening and construction of R shown for different ϵ_{str} values with a uniform grid. Results are shown for two different cross sections.

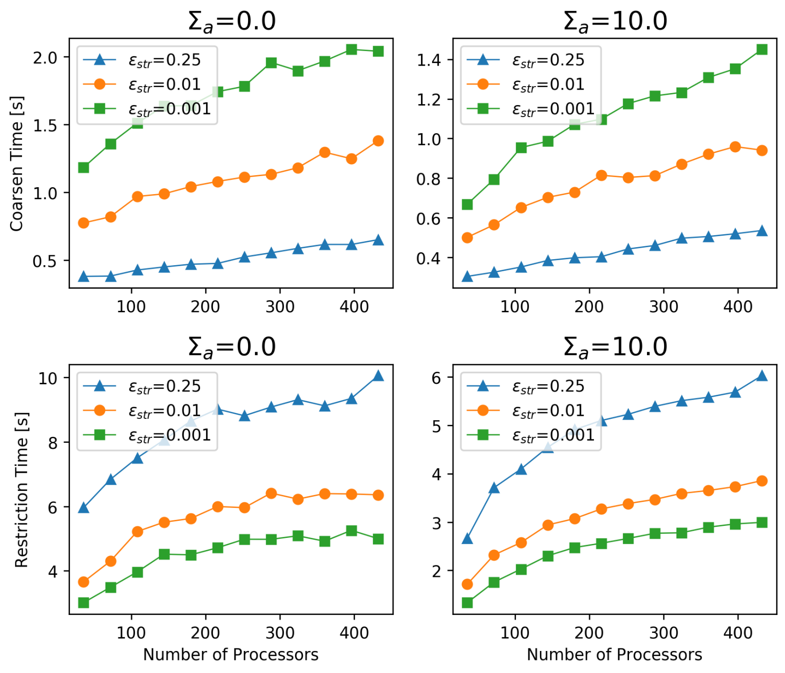


Figure 4.7: Time for coarsening and construction of R shown for different ϵ_{str} values for a grid with random vertices. Results are shown for two different cross sections.

4.2.2 3 Dimensions

For both the uniform grid and unstructured mesh, the distance-2 LAIR algorithm is used with ϵ_R=0.1 while for distance-1 LAIR, ϵ_R=0.01 is used. These selections were made in an attempt to balance the convergence factor and overall cost. Figure 4.8 shows convergence rates for both distance-1 and distance-2 LAIR on a uniform grid with two different values of ϵ_{str}. For distance-2 LAIR, decreasing the value of ϵ_{str} has only a small affect on convergence factors. For distance-1 LAIR, decreasing ϵ_{str} decreases convergence factor significantly. The dependence on direction of the convergence factor is also reduced. Importantly, for a uniform grid, distance-1 LAIR performs similarly to distance-2 LAIR. Based on this result, distance-1 LAIR will be used for the scaling tests presented in Section 6.

Figure 4.9 shows the time spent coarsening and building R for the results presented in Figure 4.8. As was the case in two-dimensions, decreasing the ϵ_{str} value increases the coarsening time and decreases the time spent building R. However, the increase in the time spent coarsening is much greater. Based on this result ϵ_{str} is set to 0.1 for the scaling test results presented in Section 6 as a balance between the improving convergence factors and the increase in coarsening time.

Based on the results shown for the uniform grid, using a very small ϵ_{str} is not practical in three-dimensions because of the increase in coarsening time. Figure 4.10 shows convergence factors for tests with an unstructured grid. For the unstructured grid, the value ϵ_{str} may be less important. The convergence factors for all tests with distance-2 LAIR are reasonable. A detailed investigation into whether distance-1 LAIR or distance-2 LAIR is optimal for the unstructured mesh is not performed. Distance-2 LAIR will be used for the scaling test discussed in Section 6.

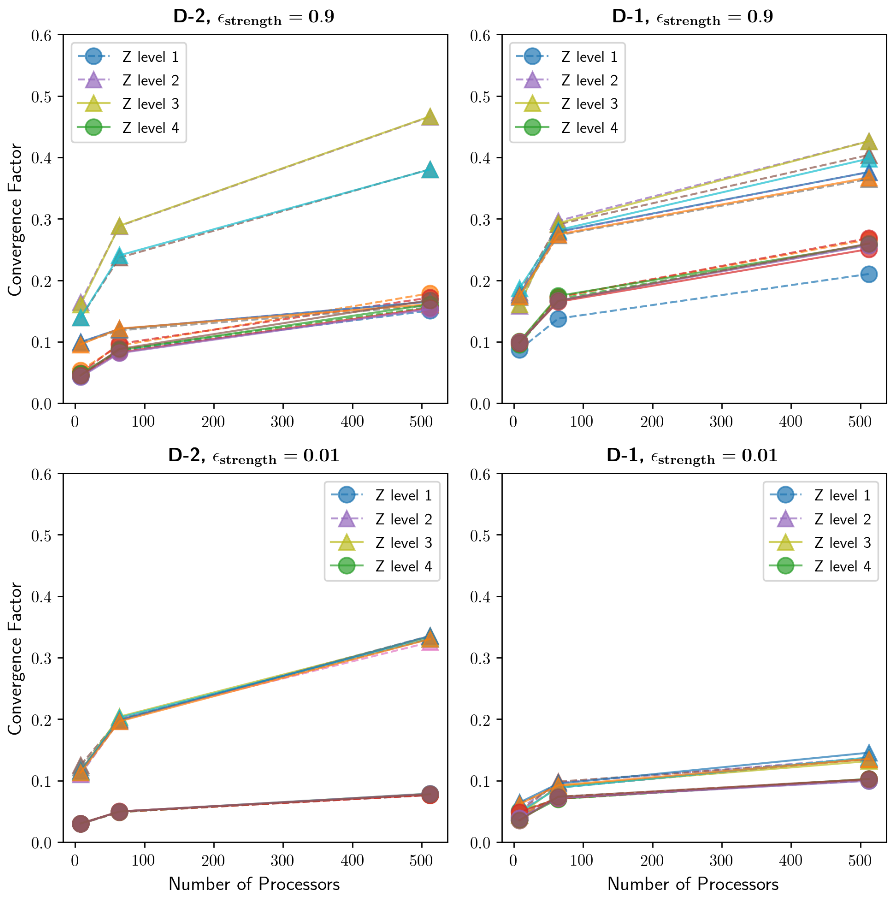


Figure 4.8: Average convergence factors for two different ϵ_{str} values with a uniform grid. Each plot shows results for 16 different directions. All directions with points in the same plane perpendicular to the z-axis have the same marker shapes and line styles.

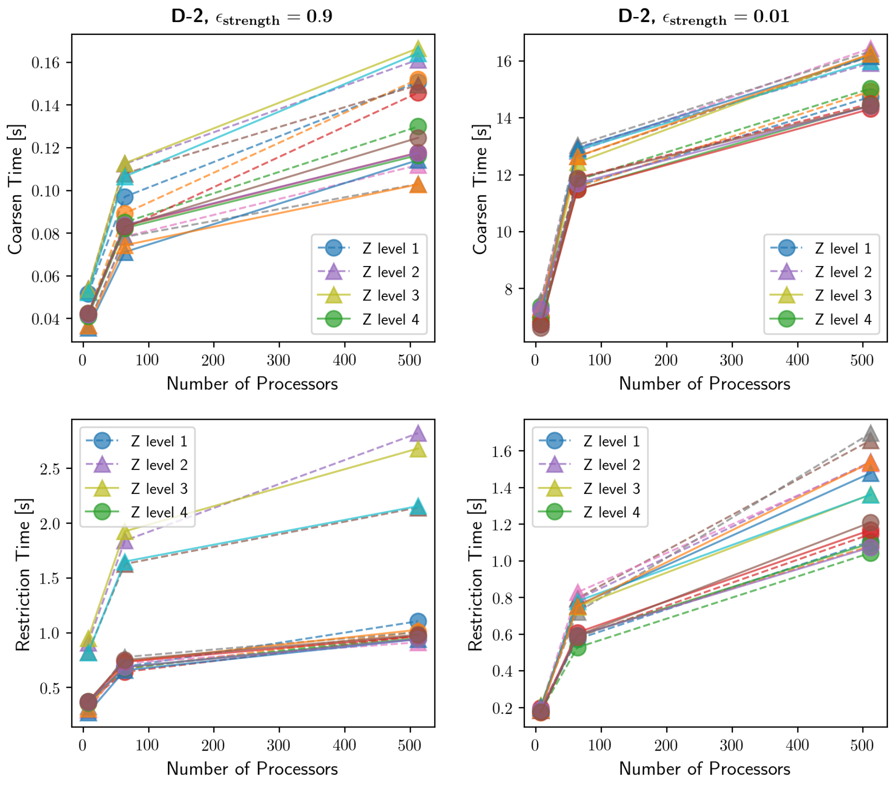


Figure 4.9: Time taken to coarsen and build the restriction operator shown for two different ϵ_{str} values. Each plot shows results for 16 different directions. All directions with points in the same plane perpendicular to the z-axis have the same marker shapes and line styles.

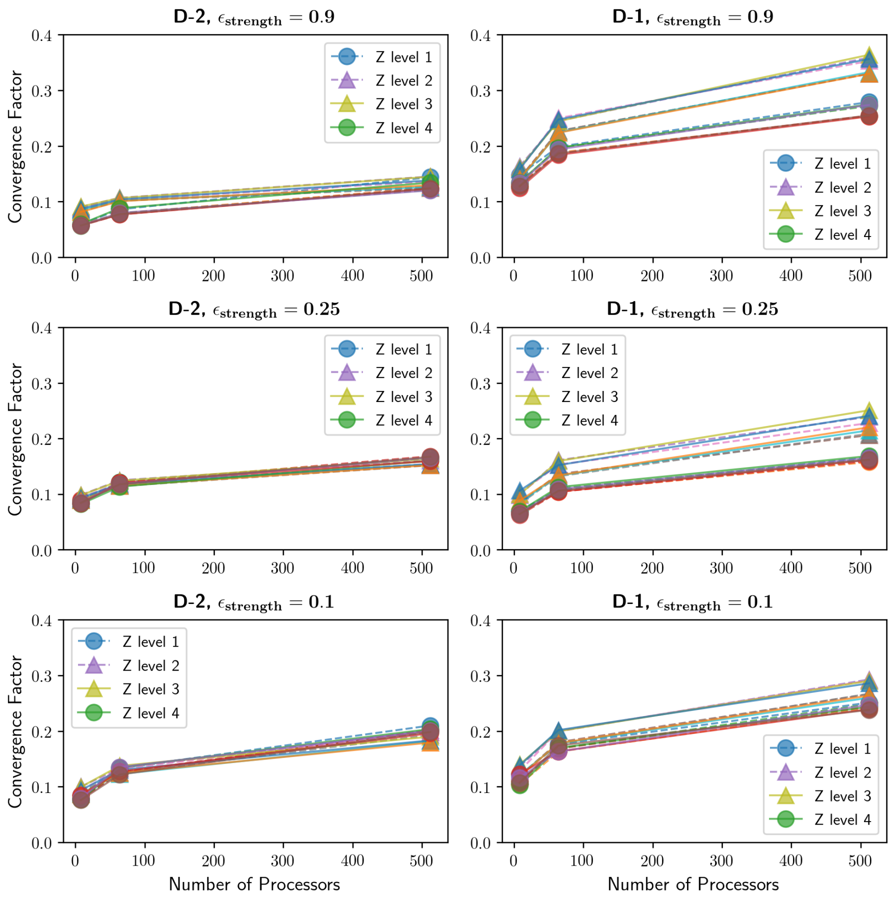


Figure 4.10: Average convergence factors for three different ϵ_{str} values with an unstructured mesh. Each plot shows results for 16 different directions. All directions with points in the same plane perpendicular to the z-axis have the same marker shapes and line styles.

4.3 Stencil Growth

As the average number of non-zero row elements in the coarse grid operator grows, the cost of calculating R will generally grow. Preventing stencil growth is a general area of interest AMG method development [17]. The magnitude and nature of stencil growth is affected by several different factors. For example, for very large values of the SOC parameter ϵ_R limit stencil growth; however, for smaller values needed for reasonable convergence rates, stencil growth does not vary significantly. This is shown in Figure 4.11.

All previous results presented in this section were generated from simulations using Falgout coarsening. *hypre* allows the user to select from several different coarsening algorithms. Figure 4.12 shows the stencil growth for the same problem solved with different coarsening algorithms. In each simulation the coarsening SOC parameter, ϵ_{str}, was the same. Clearly, the coarsening algorithm can have significant effects on the overall performance of AMG. This, however, is not investigated in detail as part of this thesis.

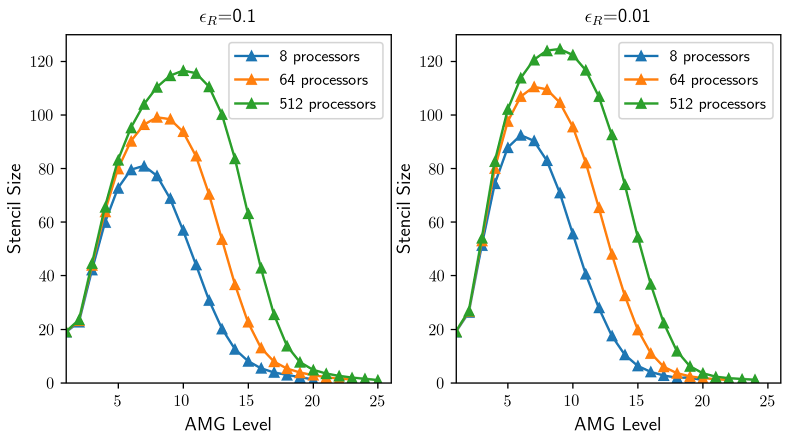


Figure 4.11: Stencil growth 8, 64, and 512 processors in a weak scaling test shown for two different ϵ_R values.

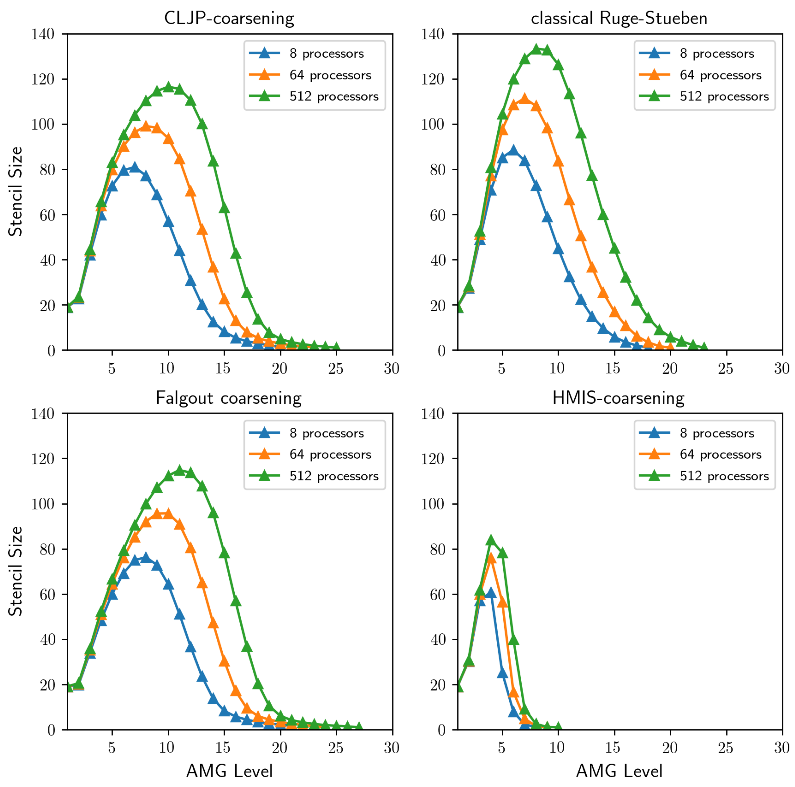


Figure 4.12: Stencil growth 8, 64, and 512 processors in a weak scaling test shown for four different coarsening algorithms.

4.4 Timing

It was shown that as the SOC parameter for coarsening was decreased, the coarsening time generally increased and the time spent building R decreased. However, the time spent building R is always a significantly expensive step. Representative timing results for these phases along with building the interpolation operator P and the coarse grid operator RAP as presented in this section. Figure 4.13 and Figure 4.14 show timing results from a uniform grid and an unstructured grid respectively. For the set of parameters used to generate these results, the time taken coarsening is significantly less than the time taken to build the interpolation operator; however, coarsening can be similarly expensive to building R depending on the coarsening parameters used.

The interpolation operator used in the pAIR algorithm is described in Section 1.4.3. Simple point-wise interpolation is used. The results shown in Figure 4.13 and Figure 4.14 are representative in general. Building the interpolation operator takes a negligible amount of time compared to the the other phases of the setup process.

Both figures presented in this section show that building the coarse grid operator RAP takes a significant amount of time. This is also a general result that is true for a variety of simulations and parameter settings.

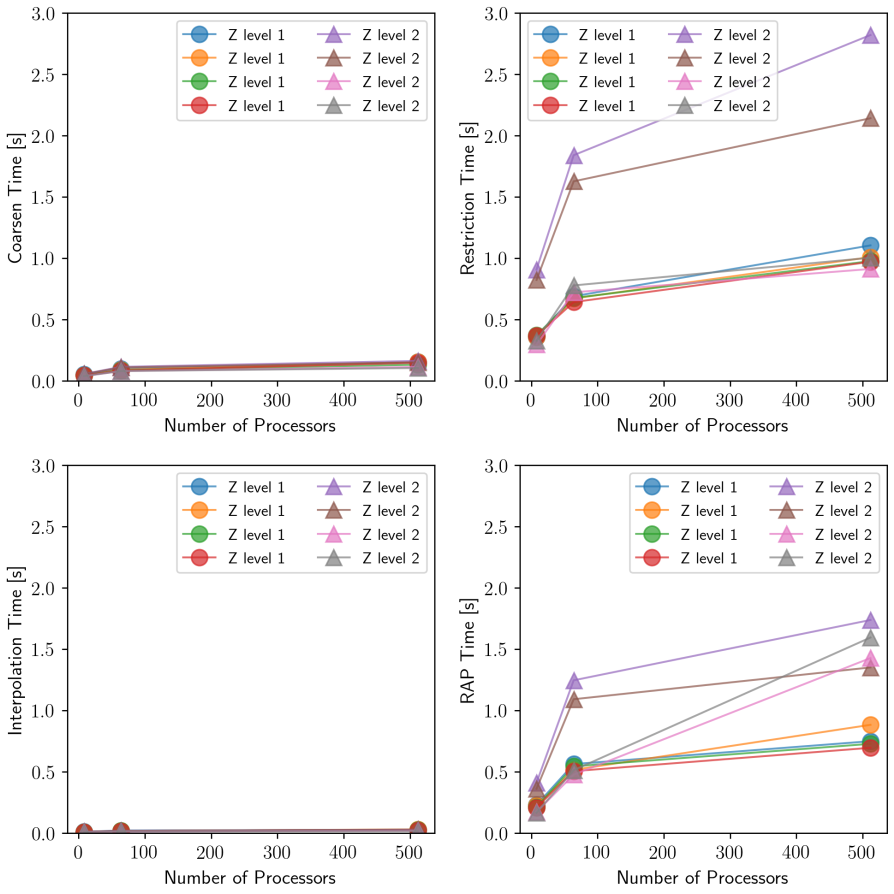


Figure 4.13: Timing for the four phases of the pAIR setup process shown for a uniform grid for different directions in two x-y planes.

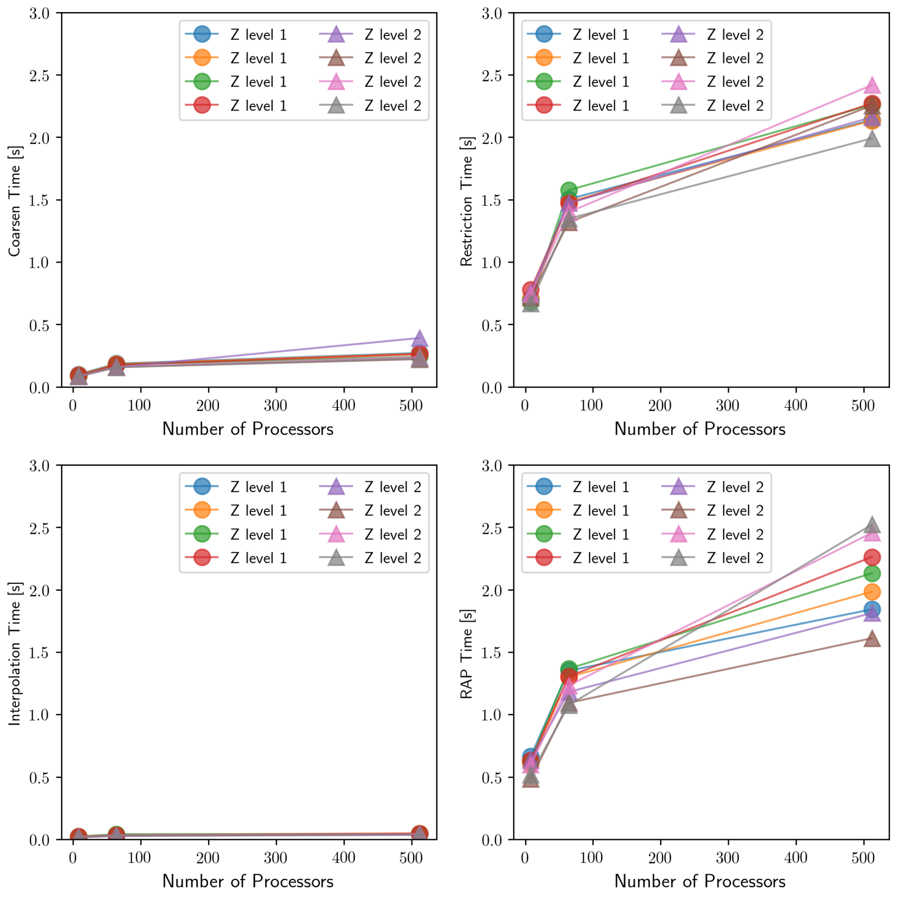


Figure 4.14: Timing for the four phases of the pAIR setup process shown for a unstructured mesh for different directions in two x-y planes.

4.5 Conclusion

As the value of ϵ_{str} is decreased, convergence factors improve, however, building and using R becomes more expensive. An initial investigation into parameter values was presented to provide some basis for selecting a parameter for scaling tests. Additionally, the use of distance-1 and distance-2 LAIR was investigated. It was determined that distance-1 LAIR would be sufficient for the uniform grid and distance-2 LAIR would be used for the unstructured grid scaling test. Coarsening parameters were shown to have a significant impact on the performance of pAIR although no attempt was made to find optimal coarsening parameters.

5. pAIR PARALLEL PERFORMANCE INVESTIGATION

Previous sections showed that building the restriction operator R as well as building the coarse grid operator RAP were always significantly more expensive than building the interpolation operator P. Depending on the parameters used, coarsening could be similarly or more expensive. However, the novel component of the current algorithm is the building of R, and therefor also RAP, so the parallel performance of these two steps is investigated in detail in this section. First, the some basic concepts related to the distributed storage and manipulation of matrices and vectors in *hypre* are reviewed.

5.1 Parallel Distributed Objects in *hypre*

The matrices and vectors used in the pAIR algorithm by *hypre* are distributed across multiple processors. The compressed sparse row data structure in *hypre* is predominately used and will be introduced briefly here, but is described in more detail in [18]. Each processor owns a continuous span of rows. For the purpose of matrix-vector multiplication, the row partitioning can be extended to a vector. Then each processor can perform a portion of a matrix-vector multiplication without communicating to other processors. The columns of the matrix that can be multiplied without communication are the rows of the vector also owned by the processor. This portion of the matrix is referred to as the diagonal portion in *hypre* and it has its own distinct data structure. To multiply columns outside the range of locally owned range of rows, a processor must communicate with neighbors to get the values of off-processor vector elements. This portion of the matrix on each processor is referred to as off-diagonal in *hypre* and it has its own distinct data structure.

A communication package is an object in *hypre* that contains the elements required for matrix-vector multiplication of distributed matrices and vectors. Each processor needs to know which processors to contact to attain the data for multiplication of the off-diagonal matrix elements. This object is described in detail in [19].

This information is relevant for building R. First, the object that marks each variable as either a C-point or an F-point is a distributed vector. A processor knows only about the C/F splitting for the diagonal portion of the matrix rows it owns. Communication must occur for each processor to receive the C/F splitting for the off-diagonal portion of its matrix rows. When distance-1 LAIR is used, the only F-points considered for any C-point are those F-points that reside on the same row as the C-point. The algorithm for computing R is described in Section 1.4.4. To compute the weights in the i^{th} row of R corresponding to i^{th} C-point, the coefficients in rows above the i^{th} row of A are required. These rows may be stored on adjacent processors and so these coefficient values must be communicated. For distance-2 LAIR, for the i^{th} C-point, F-points not in the i^{th} row, but in the row of a distance-1 neighbor are also considered, and so the amount of data that must be communicated will generally be greater.

5.2 Construction of R

For the purpose of analysis, the algorithm for computing R is broken into three granular phases:

1. Each processor first determines from which other processors it needs data and to which processors it must send data. This data is communicated

2. Each processor has all of the data it needs to build R and constructs its portion of R. This step involves local computations

3. After all the rows of R have been computed by each processor individually, a communication package must be constructed by which each processor learns its communication pattern for the new matrix R

5.2.1 Initial Assessment

An initial test was performed using 144 processors or 4 full nodes of the machine used for these tests. Figure 5.1 and Figure 5.2 show three metrics for building R with distance-2 and distance-1 LAIR respectively. These figures show the time spent by each processor on local computations and the time spent communicating data to other processors.

Figure 5.1 shows that for distance-2 LAIR, the relative importance of communication between processors decreases as the problem size per processor shrinks. This effect is less significant for distance-1 LAIR. For some parameter settings, the time spent on local computations quickly dominates the time spent in communication. For example, when ϵ_R is decreased to 0.01, most connections are considered when building R. This would tent to increase both the amount of data that must be communicated and the amount of local computations, but the growth in local computation time clearly dominates even when each processor has a relatively small domain. However, use of such a small ϵ_R is not practical in general as demonstrated by the growth in computational time. As the domain size per processor increases over 200,000 DOFs, the total time taken to construct R is approaching 20 seconds.

The results presented in this section indicate that when distance-2 LAIR is used with domain sizes per processor that are large, the time spent in local computations will generally dominate the time spent communicating data. However, communication times are of interest as well because as the problem size per processor is small, which it may be when solving a problem in practice, the communication time and computation time are similar. Additionally, for distance-1 LAIR, communication time remains significant as the problem size per processor grows. The same metrics presented in this section are also presented in the next section, but for small weak scaling tests.

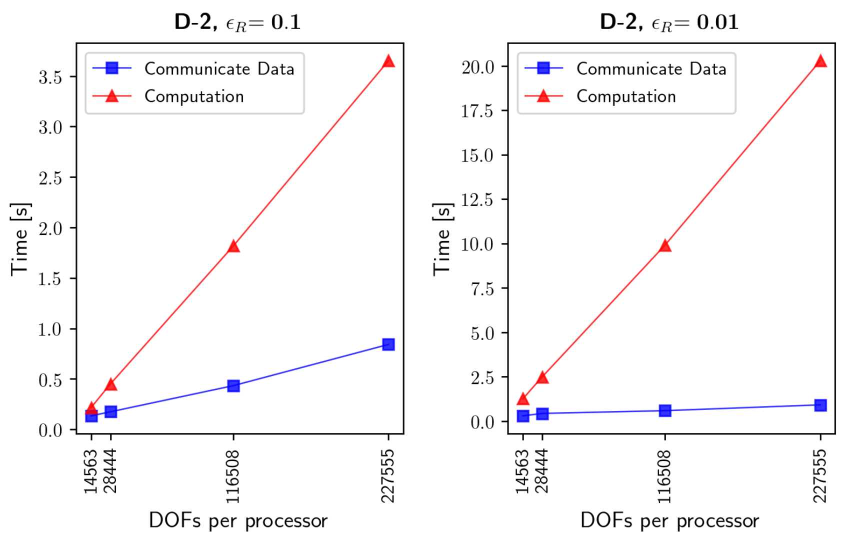


Figure 5.1: Comparison of the time taken to communicate the data required to build R with the time spent in local computation. D-2 LAIR is shown for two different ϵ_R values with a uniform grid.

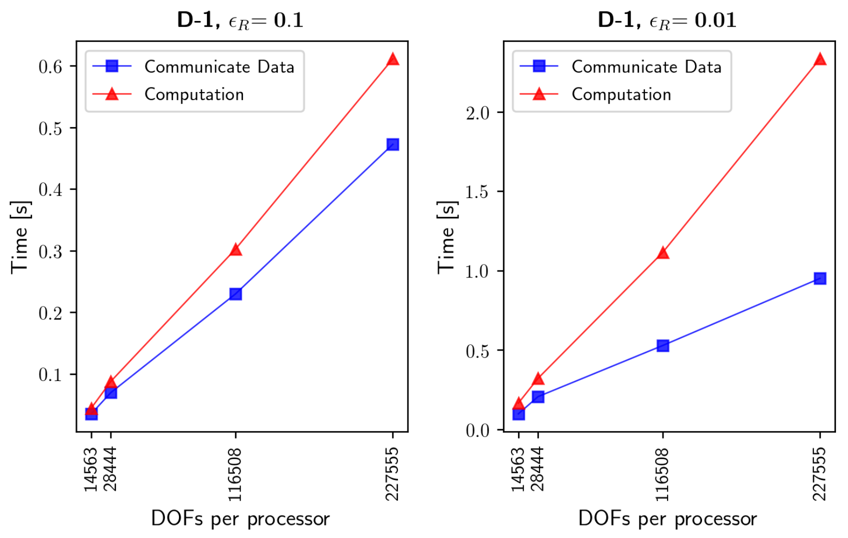


Figure 5.2: Comparison of the time taken to communicate the data required to build R with the time spent in local computation. D-1 LAIR is shown for two different ϵ_R values with a uniform grid.

5.2.2 Weak Scaling Test

Figure 5.3 and Figure 5.4 show scaling results for the three metrics investigated in the previous section. Here, the two situations of practical interest are investigated in more detail. In particular, distance-1 LAIR with a small ϵ_R parameter value and distance-2 LAIR with a larger parameter value.

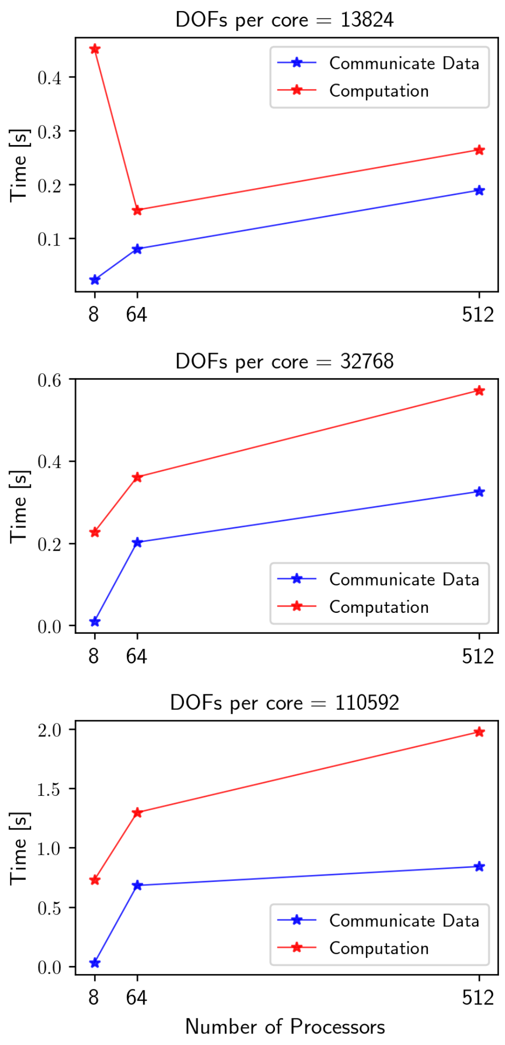


Figure 5.3: Weak scaling results comparing the time taken to communicate the data required to build R with the time spent in local computation. D-2 LAIR is shown with $\epsilon_R = 0.1$.

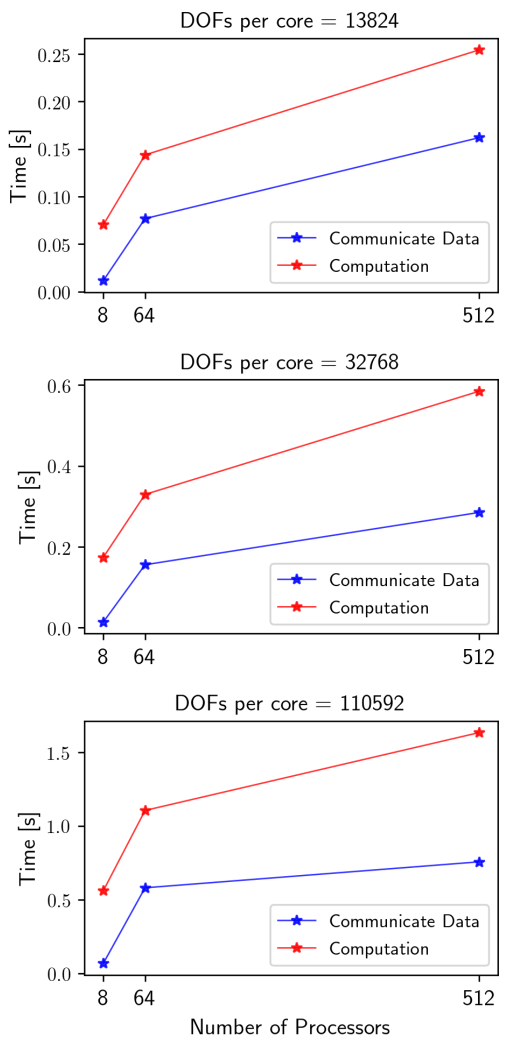


Figure 5.4: Weak scaling results comparing the time taken to communicate the data required to build R with the time spent in local computation. D-1 LAIR is shown with $\epsilon_R = 0.01$.

5.3 Detailed Profiling

Detailed profiling results for a single test are presented in this section. A single test run on 512 processors is investigated. For this test, distance-2 LAIR was used with $\epsilon_R = 0.1$ and the problem size was 35,000 DOFs per processor.

5.3.1 Detailed Profiling Results for Local Computations

Figures 5.5 through 5.7 show the time spent in local computations for the first 12 levels of the multigrid hierarchy. Figure 5.5 shows that initially, the computations are well balanced. An essential factor in the balance of computations is the mesh partitioning. For the tests presented here, the mesh was balanced across the 512 processors such that each processor had the same number of total cells. Despite the balanced mesh partitioning, Figure 5.5 shows that by level 4, a significant imbalance has developed. Additionally, the total time spent in computations is increasing from level 1 to level 4. This is not necessarily intuitive since each subsequent level should have fewer variables than the previous.

Figure 5.6 shows that the imbalance in local computation work persists and the maximum time spent in computations continues to increase until level 7. Note that the trend for level 7 looks different from the other plots because the processor ordering for all of the plots was set by sorting the time for so that level 7 was monotonically increasing. This was arbitrary and the processor ordering could have been set in different ways for plots, for example by sorting the timings for a different level. Figure 5.7 shows that by level 11, the time spent in local computations becomes insignificant.

The imbalance in local computations as well as the growth in time to build R are both of interest. First the growth in time to build R will be investigated. Figure 5.8 shows the time spent building R for each level in the multigrid hierarchy compared with the stencil size for that level. The stencil size is the number of non-zeros in the operator divided by the number of rows. Stencil growth is clearly a driving factor in the growth of computation time. This is reasonable because as the stencil grows, more points must be considered when building R.

The imbalance that develops in the local computations also appears to be related to the stencil growth. Figure 5.9 shows the local computation times for level 6. Also shown is, for each possessor domain, the ratio of cells on the exterior boundary to total cells. Clearly, for the domains with significant numbers of boundary cells, the computation time is not increasing significantly. The fact that stencil growth is truncated near exterior boundaries could explain this phenomenon. As a problem becomes bigger, the effects related to the exterior boundary should be diminished. It is then expected that larger problems would be relatively better balanced.

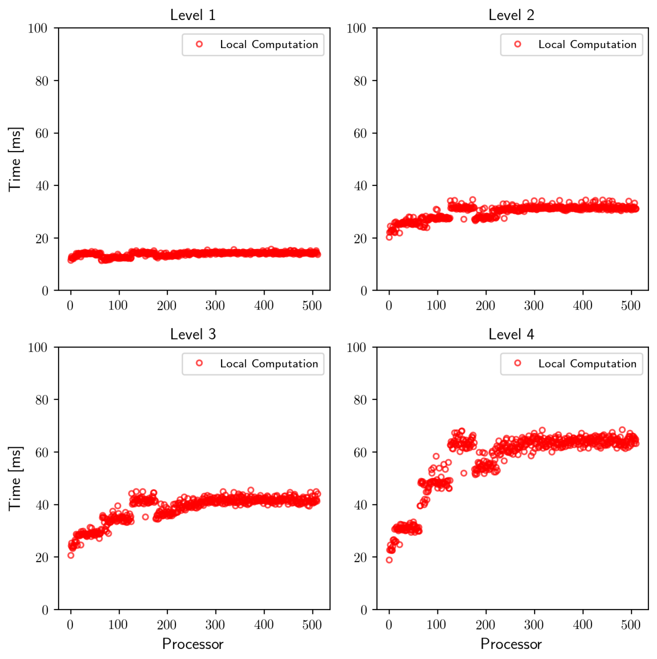


Figure 5.5: Time spent on local computations to build R for levels 1 through 4 shown for each processor.

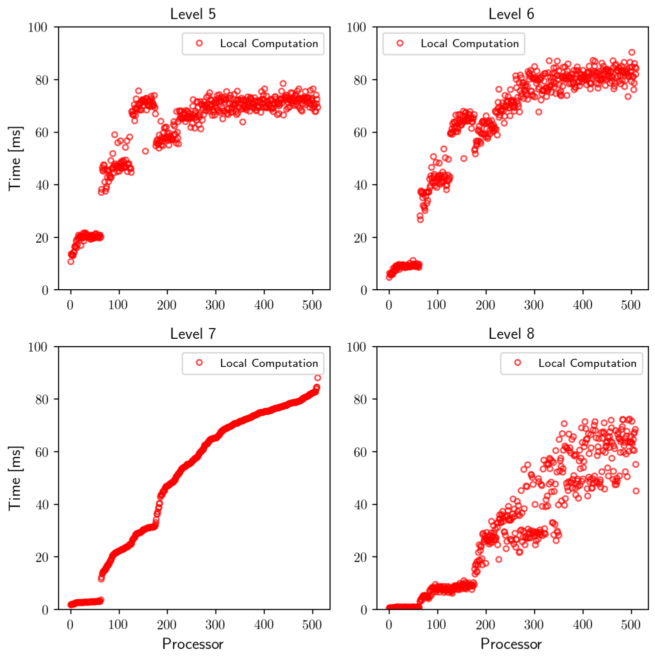


Figure 5.6: Time spent on local computations to build R for levels 5 through 8 shown for each processor.

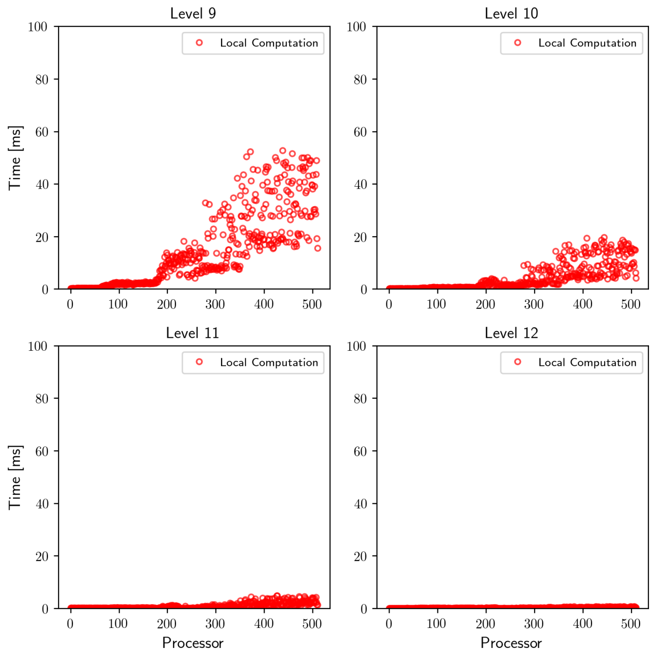


Figure 5.7: Time spent on local computations to build R for levels 9 through 12 shown for each processor.

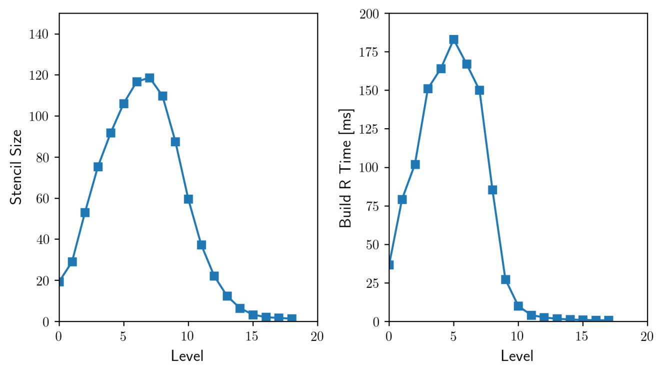


Figure 5.8: Stencil size for each level in the multigrid hierarchy compared with the time taken to build R for that level.

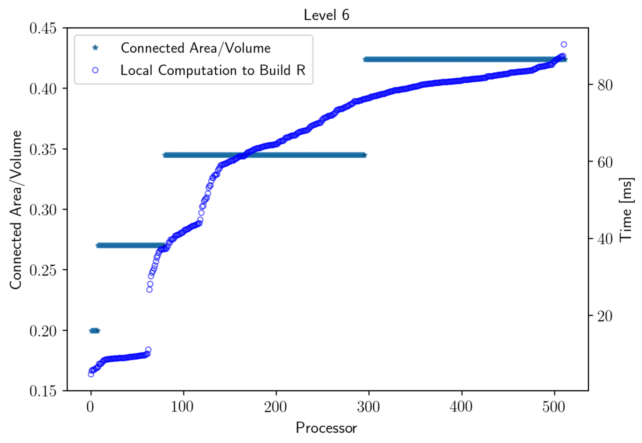


Figure 5.9: Surface area to volume ratios for each processor domain compared with the time taken to build the communication package for R for level 6 in the multigrid hierarchy.

5.3.2 Detailed Profiling Results for Local Communications

Figure 5.10 and Figure 5.11 show the time spent by each processor communicating the data required to build R. Also shown in the these plots is the time spent in local computations which is the same metric presented in the previous section. The growth in maximum time spent in communications is similar to the time spent in local computations. However, there are some clear differences. First, most processors are communicating quickly. These processors are clustered near zero in the plots. Secondly, there is a clear trend when examining the local computation times. However, there is no apparent trend in which processors are taking a long time to communicate.

In Figure 5.12 the communication time on each level is plotted by sorting the time for level 6 and then using that same processor ordering for each of the other levels. When this was done with communication times, there were clear trends in which processors were taking the most time between each level. However, Figure 5.12 does not show any clear relationship between each of the levels. In Figure 5.13 the times to communicate are sorted for each plot individually. Note then that the processor ordering is different between each plot and so processors 400-500 for any particular plot are different from those same processor number in any other plot. When the times are sorted individually, there are obvious similarities. One plausible explanation for this behavior is network saturation.

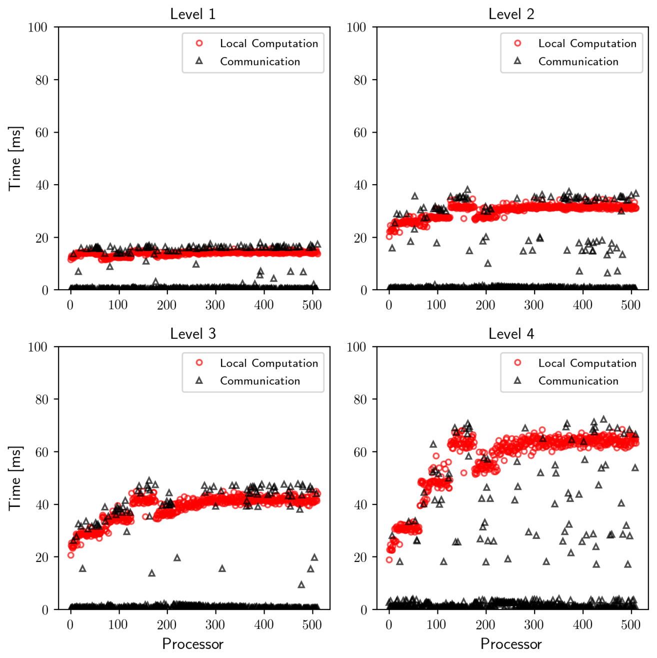


Figure 5.10: Time spent communicating data to build R compared with local computations to build R for levels 1 through 4 shown for each processor.

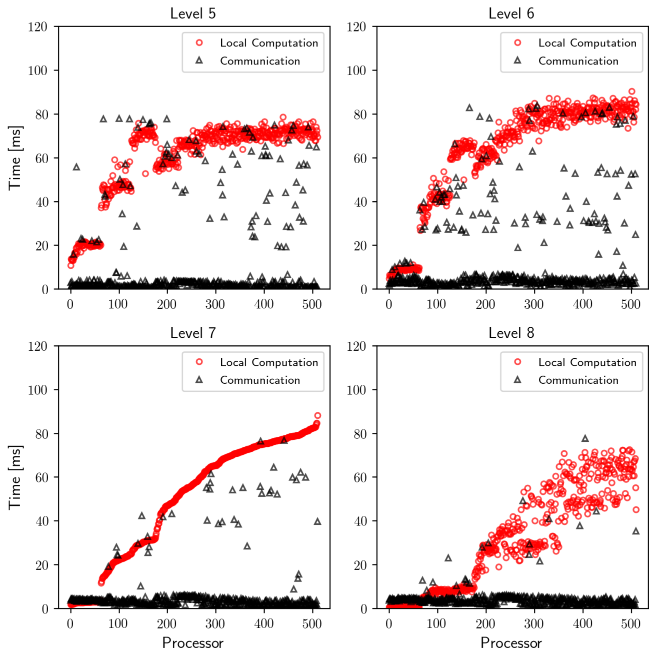


Figure 5.11: Time spent communicating data to build R compared with local computations to build R for levels 5 through 8 shown for each processor.

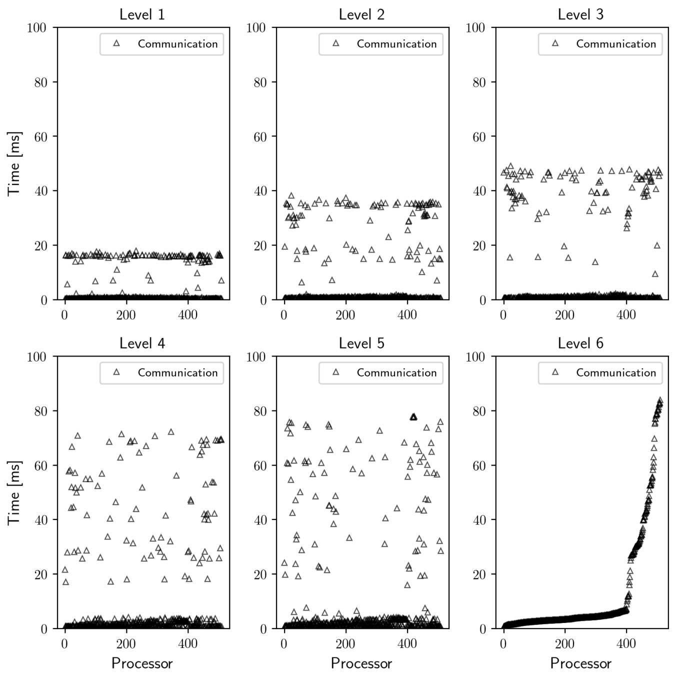


Figure 5.12: Time spent communicating data to build R for levels 1 through 6 shown for each processor. Each plot has the same ordering of processors.

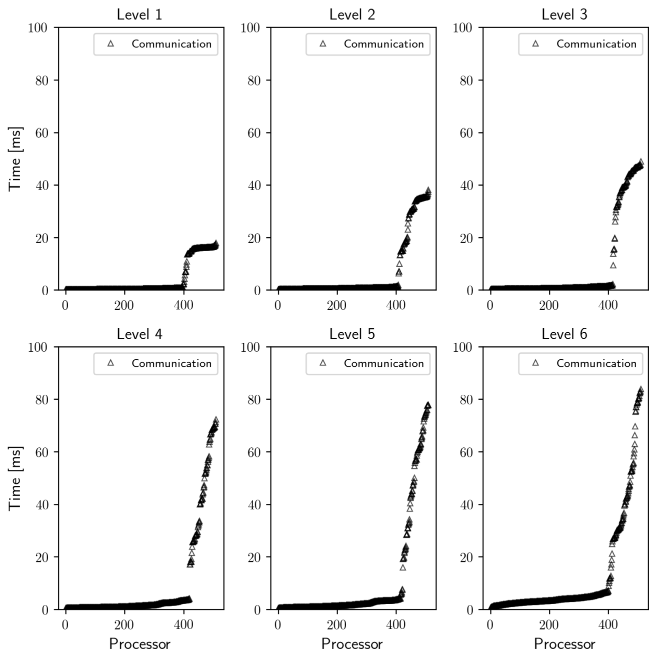


Figure 5.13: Time spent communicating data to build R for levels 1 through 6 shown for each processor. The processor order is different between each plot.

5.3.3 Testing Consistency of Communication Times

Figure 5.12 did not show any apparent relationship between the AMG levels for which processors were taking significant time to communicate. To further investigate this, the same exact test input was run four times in a row. This was done in a single batch file so that the same exact computation node configuration was used for each test. Figure 5.14 shows the time spent in local computations for each test. The processors are ordered by sorting the value for a single test and then each plot uses the same processor ordering. This demonstrates that the MPI processor number was consistent between each test.

In Figure 5.15, the time spent in local computations is shown. Again, the values in a single test were sorted and this processor ordering was used for each of the other plots. Unlike the computation time, different processors appear to be taking more time to communicate between each test. There does appear to be some similarities between the four tests. Although the trends are less similar then for local computation time, in each plot there is a similar trend for each of the processors that communicate quickly. That is a slow nearly monotonic increase in time. Also in each plot, there is a cluster of processors taking a long time in the approximate processor rank range of 450-512.

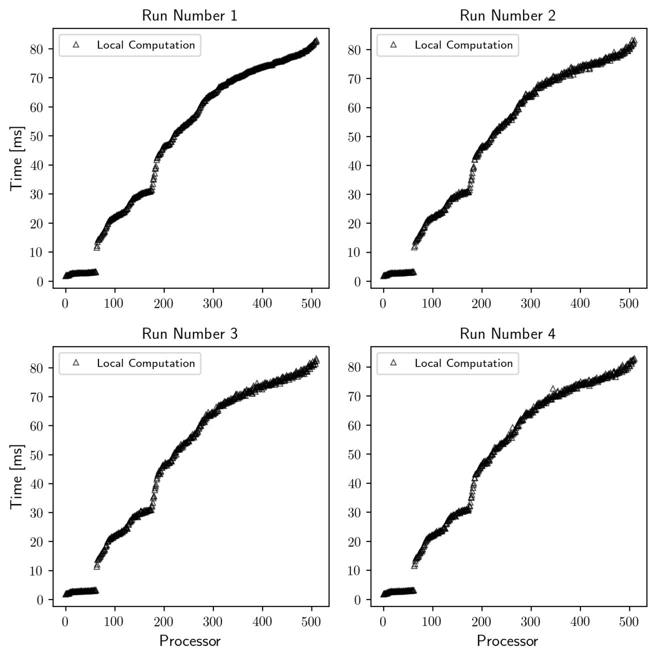


Figure 5.14: Time spent on local computations to build R for each processor.

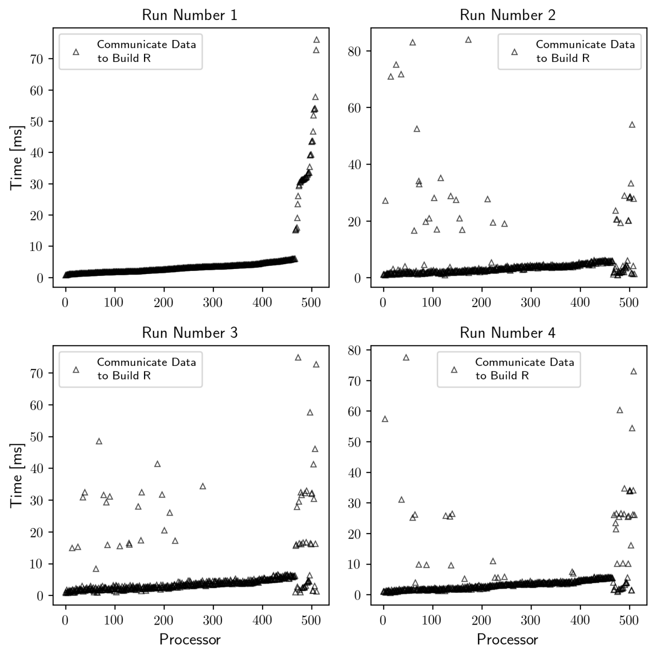


Figure 5.15: Time spent communicating data to build R for each processor.

5.4 Conclusion

The performance profiling results for building the restriction operator provided some basis for understanding the relative costs of communications versus computations. For distance-1 LAIR, communications remain an important portion of the overall cost even out to domain sizes of 227,555 DOFs per processor. For distance-2 LAIR, it is reasonable to expect both communication and computational costs to increase over distance-1 since more neighbors are considered. The results showed that for distance-2 LAIR, the importance of communication decreases more rapidly for larger domain sizes.

When local computation times were investigated in detail, two clear trends were apparent. These were that an imbalance in the local computation times developed and that the time required to build R increased until level 7. These results were well explained by growth in the stencil size and by the relative amount, for each processor domain, of exterior boundary cells to total cells. Communicating the data required to construct R also takes considerable time. On each level, the majority of processors communicate relativity quickly, but a minority get "stuck" and take a very long time. No correlations discovered explaining the patterns seen in the time required to communicate. One plausible explanation is that the network becomes saturated with messages. The pAIR algorithm is characterized by periods with mostly local computations followed by tremendous bursts in communication of data.

6. SCALING AND COMPARISON WITH PDT

A single iteration of Equation 1.4 involves solving all directions in the angular quadrature set for the previous scalar flux iterate. A single iteration is solved in weak scaling tests presented in this section. This is implemented by solving each direction individually in serial. That is, for each direction, a finite element system is setup and solved with pAIR. Thus this implementation is parallel over the spatial dimension only.

Towards the bottom of the multigrid grid hierarchy, there are few variables compared to the original grid. The coarsest grid at the bottom of the hierarchy is generally small enough that a direct solver can be used effectively. The coarser grids towards the bottom of the hierarchy are important in determining the parallel scalabity of multigrid methods. A large number of processors cannot be used efficiently on the coarser grids since there are not enough variables. Thus parallel scalabity is a function of the total number of coarse grids and the number of times computations are performed on these grids. The number of coarse grids should grow logarithmically with the total problem size. Additionally, for the V-cycle multigrid results presented here, the coarser levels are visited one time per cycle. Thus, for an ideal multigrid method, the increase in solution time should grow linearly in loglog space with a slope equal to one when plotted with the logarithm of total problem size or equivalently the total number of processors in a weak scaling test. In practice, factors such as growth of the convergence factor and coarse-grid fill in can lead to a exponent m, corresponding to $log^m(P)$, that is greater than one.

6.1 Uniform Grid

Figure 6.1 shows the results of a weak scaling test performed with a uniform mesh and fixed spatial domain size of 32768 degrees of freedom per processor. This size was set by starting with a uniform coarse mesh of 8 total cells. The mesh was distributed across 8 processors and then 4 uniform refinements were performed. For each subsequent test, the mesh was refined N additional times and the problem distributed over $8 * 8^N$ processors. The calculations were performed with a square Chebyshev-Legendre S_4 quadrature set having 32 directions [13].

Figure 6.1 provides initial evidence for the scalability of the pAIR algorithm. For both the solution phase and setup phase, the growth in total time shows a relatively linear trend in loglog space when plotted with the logarithm of the number of processors.

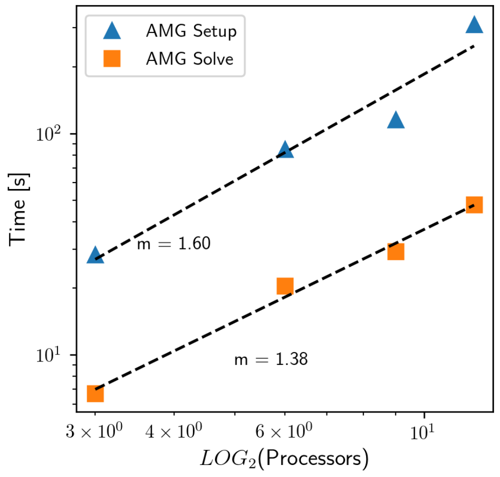


Figure 6.1: Scaling test results showing both the setup time and solve time with a uniform grid. The reported time is the total time to solve 32 directions.

The same structured grid problem was solved using PDT. The result presented in Figure 6.2 can be compared with the structured grid result shown in 6.1. However, this is a rough comparison. PDT is a highly optimized program while the program constructed to test pAIR is not optimized. Additionally, performance improvements are also being made to pAIR at this time. Despite the roughness of the comparison, these initial results demonstrate that parallel sweeps will generally be faster for a structured grid up to some number of processors not yet determined.

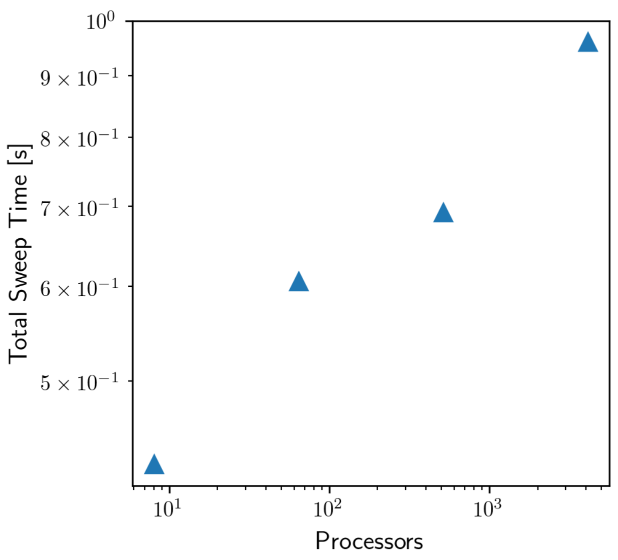


Figure 6.2: Total time to solve 32 directions with PDT.

6.2 Unstructured Mesh

Figure 6.3 shows the results of a weak scaling test performed with the unstructured extruded mesh shown in Figure 2.4. The coarse mesh contained 6370 cells. It was uniformly refined after being distributed over 8 processors, leading to a spatial domain size of 50960 degrees of freedom per processor. The same angular quadrature used for the uniform grid was used for the unstructured mesh.

The result for the unstructured mesh looks similar to the that for the structured grid. Again, this result provide initial evidence for the scalability of the pAIR algorithm.

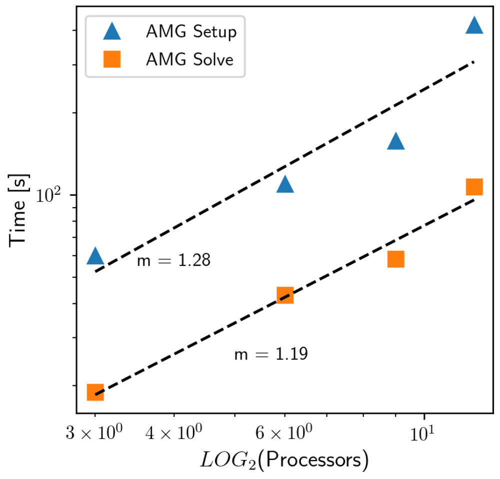


Figure 6.3: Scaling test results showing both the setup time and solve time with a unstructured mesh. The reported time is the total time to solve 32 directions.

7. pAIR LAIR MODIFICATION

7.1 Introduction

The results in previous sections showed that building R was often an expensive part of the setup phase. Construction of R can also have a significant impact on convergence. Using smaller values for ϵ_R results in better convergence. As the value of ϵ_R is lowered, more connections are considered when building R and so the floating point operations associated with the linear solves required to determine the restriction weights increases. The effect of this was demonstrated in Section 4.1

Filtering R after construction for small values can help limit the growth of nonzeros. But each small value is still computed before being filtered. Previous sections showed that both distance-1 and distance-2 LAIR become more expensive as the value of ϵ_R is made small. However, distance-2 becomes more expensive more quickly. A possible method for mitigating the increased cost would be considering two different ϵ_R parameters, ϵ_{R1} and ϵ_{R2}, where the ϵ_{R1} parameter is used for distance-1 connections and the ϵ_{R2} parameter is used for distance-2 connections. The value of ϵ_{R1} can be made very low to consider most immediate neighbors and ϵ_{R2} can be set to some larger value to limit the number of distance-2 neighbors being considered, but still get some convergence improvement.

7.2 Initial Test Results

Use of two SOC parameters was implemented in *hypre* and several tests were performed. Figure 7.1 show results from one such test. The values of ϵ_{R1} and ϵ_{R2} tested are shown in the figure titles. This initial result is representative and the use of two different SOC parameters appears to have the potential to reduce the over pAIR setup time. The next step in determining the usefulness of this algorithm improvement will be to test the use of two different SOC parameters against simply using a single larger SOC parameter in a variety of situations.

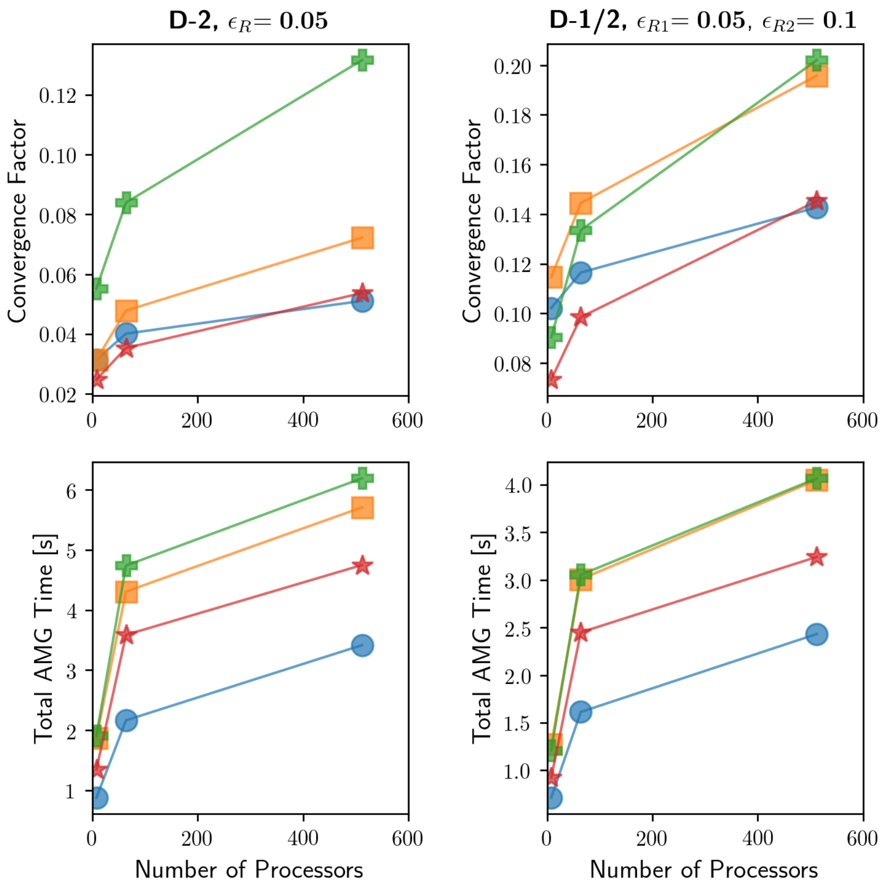


Figure 7.1: Average convergence factors and total AMG time compared for D-2 LAIR and the modified D-1/2 logic. Results are shown for four different directions with a uniform grid.

8. CONCLUSION

In this paper, the pAIR solver is shown to be effective and scalable for solving the source iteration equations of the S_N approximation to the transport equation. Because pAIR is an algebraic solver implemented in a popular linear algebra library, it can be easily interfaced as a black-box solver to other programs. As demonstrated here, a program capable of solving transport was created with a popular FEM library. This is not generally possible to do with parallel sweeps, which even on structured grids, require specialized communication scheduling logic that is integral with domain partitioning. Using parallel sweeps to solve problems on unstructured grids is more difficult and generally not as efficient. To our knowledge, no detailed performance model analogous to that presented in [5][6] has been developed for sweeping on unstructured grids. However, pAIR appears to perform similarly on both structured and unstructured grids.

Section 4 briefly investigated the importance of the coarsening and restriction models. Parameter selection can have significant impacts on performance. The primary metric of interest in this section was the total time to solution, but as was demonstrated by the stencil growth results, parameter selection will also affect total memory requirements. Reasonable parameters were selected for the scaling tests presented in Section 6. However, an exhaustive study of optimal parameters was not performed. Finding optimal parameters for different situations is an important task for future investigation.

Detailed profiling of the parallel performance of *hypre* revealed that, when building the restriction operator, both local computation and communication between processors can be significant. For distance-2 LAIR, local computations tend to dominate communication costs as the problem size per processor grows. Local computation costs and communication costs were investigated in detail. It was determined that the stencil fill-in was an important factor in the high costs for building R. An important factor in the time required to communicate data before building R appears to be that a tremendous number of messages must be passed. This burst of communication appears to cause some network saturation. Possible strategies to avoid this problem include changing the algorithm slightly so that not all processors attempt to communicate at once.

A potential algorithm improvement for distance-2 LAIR was introduced in this thesis. The improvement involved using two SOC parameters, one for distance-1 connections and a second for distance-2 connections instead of a single parameter. In initial tests, this improvement was shown to decrease the pAIR setup time. Future testing of this improvement will include using it in more situations and comparing it with other strategies for reducing the time taken to build R.

REFERENCES

[1] R. Falgout, “An introduction to algebraic multigrid,” *Computing in Science & Engineering*, vol. 8, no. 6, pp. 24–33, 2009.

[2] T. A. Manteuffel, S. F. McCormick, S. Munzenmaier, J. W. Ruge, and B. S. Southworth, “Reduction-based algebraic multgrid for upwind discretizations,” *SIAM Journal on Scientific Computing*, submitted.

[3] T. A. Manteuffel, J. W. Ruge, and B. S. Southworth, “Nonsymmetric reduction-based algebraic multigrid based on local approximate ideal restriction (lair),” *SIAM Journal on Scientific Computing*, submitted.

[4] R. S. Baker and K. R. Koch, “An Sn algorithm for the massively parallel cm-200 computer,” *Nucl. Sci. Eng.*, vol. 128, 1998.

[5] T. S. Bailey and R. D. Falgout, “Analysis of massively parallel discrete-ordinates transport sweep algorithms with collisions (llnl-conf–407968),” Oct. 2008.

[6] M. P. Adams, M. L. Adams, W. D. Hawkins, T. Smith, L. Rauchwerger, N. M. Amato, T. S. Bailey, and R. D. Falgout, “Provably optimal parallel transport sweeps on regular grids (llnl-conf–407968),” 2013.

[7] J. Liu, L. Chi, W. QingLin, and C. Gong, “Parallel Sn sweep scheduling algorithm on unstructured grids for multigroup time-dependent particle transport equations,” *Nuclear Science and Engineering*, vol. 184, pp. 527–536, 12 2016.

[8] G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmoeller, T. Heister, L. Heltai, K. Kormann, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells, “The `deal.II` library, version 9.0,” *Journal of Numerical Mathematics*, 2018, accepted.

[9] C. Burstedde, L. C. Wilcox, and O. Ghattas, "`p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees," *SIAM Journal on Scientific Computing*, vol. 33, no. 3, pp. 1103–1133, 2011.

[10] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, "An overview of the trilinos project," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, 2005.

[11] M. Sala and M. Heroux, "Robust algebraic preconditioners with IFPACK 3.0," Tech. Rep. SAND-0662, Sandia National Laboratories, 2005.

[12] S. Shende and A. D. Malony, "The tau parallel performance system," *International Journal of High Performance Computing Applications*, vol. 20, no. 2, pp. 287–311, 2006.

[13] W. Walters, "Use of the Chebyshev-Legendre quadrature set in discrete-ordinate codes," Los Alamos National Lab., NM, 1987.

[14] D. S. Kershaw, "Differencing of the diffusion equation in lagrangian hydrodynamics codes," *J. Comput. Phys.*, vol. 39, no. 375, 1981.

[15] A. J. Cleary, R. D. Falgout, V. E. Henson, and J. E. Jones, "Coarse-grid selection for parallel algebraic multigrid," in *Solving Irregularly Structured Problems in Parallel*, 1998.

[16] V. E. Henson and U. M. Yang, "Boomeramg: A parallel algebraic multigrid solver and preconditioner," *Applied Numerical Mathematics*, vol. 41, no. 1, pp. 155 – 177, 2002.

[17] H. De Sterck, U. Yang, and J. Heys, "Reducing complexity in parallel algebraic multigrid preconditioners," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 4, pp. 1019–1039, 2006.

[18] R. D. Falgout, J. E. Jones, and U. M. Yang, "Pursuing scalability for hypre's conceptual interfaces," *ACM Trans. Math. Softw.*, vol. 31, pp. 326–350, Sept. 2005.

[19] A. Baker, R. Falgout, and U. Yang, “An assumed partition algorithm for determining processor inter-communication,” *Parallel Computing*, vol. 32, no. 5, pp. 394 – 414, 2006.