Algebraic Domain Decomposition Preconditioner for Electromagnetic Computations

Tamimu Mohammed Gadafi
Email: tamimugadafi@gmail.com or 202024110101@std.uestc.edu.cn
Institution: University of Electronic Science and Technology of China
Department: Mathematical Sciences
Professor Li Liang
Email: plum_liliang@uestc.edu.cn
Institution: University of Electronic Science and Technology of China
Department: Mathematical Sciences
Kanwal Asia
Email: asiamaths@feishu.uestc.cn
Institution: University of Electronic Science and Technology of China
Department: Mathematical Sciences

Article Info
Page Number:
2077–2092
Publication Issue:
Vol. 72 No. 1 (2023)

Abstract
The solution of large-scale linear equations is a computationally challenging problem in scientific and engineering computing. Due to memory and CPU time constraints, usually only iterative solver can be used. In recent years, many solvers have been built to handle such challenging problems. In this paper, our main objective is to design a new ADDM preconditioner for finding a method of linear equations which are efficient, robust, and can also be implemented well. We designed and implemented an efficient ADDM preconditioner algebraically without having to know an information of the problem in general and also without any hypothesis on the least-squares matrix except that it is sparse, this and other results showing how our designed preconditioner outperformed other preconditioners are shown in this paper.We discussed how the ADDM preconditioner can be designed using the Matlab and Python codes and the results are analyzed and presented to show the performance of the proposed preconditioner. The ADDM preconditioner designed are very robust and efficient since it outperforms other preconditioners designed in the past in terms of convergence. The problems which could possible arise from a very practical applications are actually used to make a relation in the performance of our new preconditioner and the other related preconditioners designed in the past.

Article History
Article Received: *15 May 2023*
Revised: *24 June 2023*
Accepted: *18 July 2023*

Keywords: Algebraic domain decomposition, Additive Schwarz, Large liner systems, Linear least-squared, precondition.

Introduction

The success of large linear systems constantly play a significant function in scientific computing. The difficulty to be resolved are often a really large size which needs to be addressed by decomposing the large linear problems in to a smaller one in order to achieve a desire result. We try to look at this problem in terms of electromagnetic computations, which

primarily includes only two numerical solvers. The first solver looks at the direct computation of differential equation and the second solver deals with a derived integral equation from the Maxwell's equations. For the purpose of our paper, preconditioner is required in order to give a condition number of problems into a more desirable pattern for numerical computations. Its interest is much more in reducing the condition number of the problems through decomposition.

There have been many domain decomposition algorithms developed over the years, however, there is still lack of black-box routines working at the matrix level which could further lead to the general adoption of these techniques in both engineering and scientific computing community. As part of this paper, we want to follow the path which leads to the construction of such black-box solver by a collaboration between numerical analysis and computer science. The general objectives of this paper is to design and implement scalable preconditioner, which is the bottleneck problem for a lot of real-world engineering or scientific research. This paper seeks to design algebraic domain decomposition (ADDM) method adapting to electromagnetic (EM) computation as a preconditioner of a Krylov subspace iterative solver. Some special properties arising from linear systems of electromagnetic computations is what we want to follow in order to design an efficient algebraic domain decomposition method (ADDM) preconditioners.

In practice, every good preconditioner should satisfy so many constraints. First and foremost, it must be cheaper or affordable in terms of computation and also to apply in both memory storage and computational time. Because the interest of this is parallel applications, therefore, the designing and execution of the system's preconditioner should also be parallelizable and scalable. That is, the preconditioned iterations should always converge rapidly, and therefore the performance should not be degenerate in case the number of processors increases. There are two primary categorization of preconditioners, the first one is to design specialized algorithms that are very closed to an optimal for any narrow type of problems, whereas the second class is a general-purpose algebraic method. But however, this type of preconditioning requires an absolute complete knowledge about the problem, and this may not always be feasible. Furthermore, these problem specific approaches are generally very delicate to the details of the problem, and therefore, even if there is a small change in the problem. Sometimes the algebraic methods on settings may or may not reduce the solver's effectiveness only made use of information that contained in the coefficient of the various matrices. These techniques usually achieve a reasonable efficiency on a broader scope of problems. In general, these preconditioners are very simple to employ and also well suited for any irregular problems. Furthermore, one crucial feature of such an approach is the fact that, they can be adapted and tuned to exploit specific application.

Basic Iterative Methods

Broadly, an iterative method is in essence a mathematical procedure that uses an initial value to produce a sequence of rising an approximate solutions for a class of problems which is an n-th approximation derived from the previous ones.[11][14] We shall consider Jacobi, Gauss-Seidel, SOR, Chebychev methods and these are explained below.

The Jacobi Method, which is required for the determination of the solutions to strictly

diagonally dominant systems of linear equations, is the first fundamental iterative method to be taken into consideration. This is accomplished by solving for each diagonal element and then plugging in an approximation of its value. After then, the procedure is recurrent until convergence is reached. This algorithm simplifies the Jacobi transformation method of matrix diagonalization.[1][13] The method is described here as an iterative process.

Let $Ax = b$, be a square system of n linear equations, where :

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Then *A* can be decomposed in to a diagonal component of *D*, a lower triangular part *L* and an upper triangular part *U*:

$$A = D + L + U \tag{2.1}$$

Where, D = $\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}$ and L + U= $\begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}$

The solution is then obtained iteratively by

$$X^{(k+1)} = D^{-1}(b - (L + U)X^{(k)}), \tag{2.2}$$

Where $X^{(k)}$ is the k-th approximation or iteration of x and $X^{(k+1)}$ is the next or k+1 iteration of x. The element based formula is thus:

$$X^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}), \qquad i = 1, 2, \ldots, n. \tag{2.3}$$

Every element in the$X^{(k)}$except itself is necessary for the computation of $X_i^{(k+1)}$. We cannot replace $X_i^{(k)}$with$X_i^{(k+1)}$, unlike the Gauss-Seidel technique, because the value is required for the remainder of the computation. Two vectors of size n make up the bare minimum of storage.[4][5][7]

In general, the technique converges when $P(D^{-1}(L + U)) < 1$. Where P is the spectral radius defined as $P(A) = Max^*|\lambda_1|, \quad |\lambda_N|$+ which is the λ_1 eigenvalues of *A* as the matrix.

Again, Gauss-Seidel method is another iterative method which is sometimes referred to as the consecutive displacement method or the Liebmann method.[8] Which is necessary for solving a system of linear equations and which is much comparable to the Jacobi approach. Convergence is only achieved if the matrix is absolutely diagonally dominant, symmetric, and positive definite, even though it can be applied to any matrix with non-zero diagonal components.[21][30][34]

The Gauss-Seidel method is discussed below. A square system of *n* linear equations with an unknown *X* must be solved.

Let $Ax = b$, which is defined by the iteration:

$$L_* X^{(k+1)} = b - UX^{(k)} \tag{2.4}$$

$X^{(k)}$ represent the *k-th* iteration of *x*, and $X^{(k+1)}$ which is the next or *k+1* iteration of *x* and the decomposition of matrix A to lower triangular component L_* , and a strictly upper triangular component U i.e, $A = L_* + U$. In more details, we write out A, x and b in their component form:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

The breakdown of A into its rigorously upper triangular component and its lower triangular component is therefore provided by:

$$L_* = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

The system of linear equations may be rewritten as:

$$L_* X = b - UX.$$

Using past values for x on the right side, the Gauss-Seidel method now resolves the left side of this formula for x.[6][8][10] this may be presented as:

$$X^{(x+1)} = L_*^{-1}(b - UX^{(k)}).$$

However, by fetching benefit of the triangular form of L_* the elements of $X^{(x+1)}$ can be computed sequentially using forward substitution:

$$X_i^{(x+1)} = \frac{1}{a_{ii}} \left(b - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right)$$

The process is broadly continued until the occurrence made by a sufficiently small residual.

In addition, a faster-converging variation of the Gauss-Seidel method for resolving a linear system of equations is known as successive over-relaxation (SOR).[28][29] Any iterative process that converges slowly can be implemented using a similar technique. The motivation of this method is to improved on the Gauss-Seidel loop by taking an appropriate weighted average of the $X_{m+1,j}$ *and* $X_{m,j}$.

Now, using the same square matrix as the matrix of Gauss-Seidel method, the system of linear equations may be rewritten as:

$$(D + \omega L)X = \omega b - ,\omega U + (\omega - 1)D\text{-}X \tag{2.5}$$

For a constant $\omega > 1$, this is referred to as relaxation factor.

Analytically, this may be written as:

$$X^{(x+1)} = (D + \omega L)^{-1}(\omega b - ,\omega U + (\omega - 1)D\text{-}X^{(k)}) = L_{\omega}X^{(k)} \quad (2.6)$$

$X^{(k)}$ represent the $k - th$ iteration of *x*, and $X^{(k+1)}$ is also the *k+1* or next iteration of *x.*

Again, by fetching benefit of the triangular form of $(D + \omega L)$, the elements of $X^{(x+1)}$ can be computed sequentially using forward substitution:

$$X_i^{(x+1)} = (1 - \omega)X_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j<1} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\right) \quad (2.7)$$

$$\mathrm{i} = 1, 2, \ldots, \mathrm{n}.$$

Classic Preconditioning Techniques

Simply put, preconditioning is the process of changing the original linear system into a more manageable version that nonetheless has the same solution. In general, the quality of the preconditioner has a significantly greater impact on an iterative technique's dependability when handling different applications than do the specific Krylov subspace accelerators employed.[15][16][19]

It is important to first think about the choices for preconditioning a system. Finding a preconditioning matrix *M* is the initial step in preconditioning. The matrix *M* can be defined in many different ways but it must satisfy a few minimal requirements. Practically speaking, the cost of solving linear systems $Mx = b$ is the primary criterion for *M*. This is due to the fact that each step of the preconditioned algorithms will call for a linear system solution using the matrix *M.* Also *M* should close to *A* in some sense and it should clearly be non-singular.

Considering the ways in which the preconditioner is applied to solve the original system. Once a preconditioning matrix *M* is available, there are three known ways of applying the preconditioner. The preconditioner can be applied from the left, leading to the preconditioned system:

$$M^{-1}Ax = M^{-1}b \quad (2.8)$$

Alternatively, it can also be applied to the right:

$$AM^{-1}u = b, x \,\Xi\, M^{-1}u$$

Note that the above formulation amounts to making the change of variables $u = Mx$, and finding with regard to the unknown *u*, the system. Finally, a common situation is when the preconditioner is available in the factored form:

$$M = M_L M_R$$

where, typically $M_L$and M_R are triangular matrices. In this situation, the preconditioning can be divided

$$M_L^{-1}AM_R^{-1}u = M_L^{-1}b, x \,\Xi\, M_R^{-1}u$$

It is imperative to preserve symmetry when the original matrix is symmetric, so the split preconditioner seems mandatory in this case. However, there are other ways of preserving symmetry, or rather to take advantage of symmetry, even if *M* is not available in a factored form.[17][18]

Domain Decomposition Methods

In preconditioning methods, domain decomposition refers to the process of subdividing the solution of the large linear system into a smaller problems and therefore, its solutions can actually be used to produce a solver (or preconditioner) for the system of equations that effect from the discretization of the PDE on the whole domains. In this context, domain decomposition refers only to the solution method for the algebraic system of equations arising from discretization.[40][45]

There are some specific importance which is given to the techniques which are well appropriate to the parallel method of large-scale scientific applications and progressive numerical simulations. Some computational aspects related to their parallel implementation are also addressed. This thesis is not purposely only for those who specialized in domain decomposition but rather for scientists who also have some special knowledge about linear algebra and discretization techniques and those who would like an introduction into domain decomposition.[49]

Two-level domain decomposition method

Single level methods are effective only for a small number of subdomains. The problem with single level methods is that the information about f and g in (2.13) in one subdomain traverse through all the intermediate subdomains only through their common interfaces.[36][42] Thus for example the convergence rate of the single level adaptive Schwarz method becomes progressively worse when the number of subdomains becomes large. From an algebraic point of view this loss of efficiency is caused by the presence of small eigenvalues in the spectrum of the preconditioned, coefficient matrix. They have a harmful influence on the condition number, thus in addition to traditional preconditioner like ASM, a second kind of preconditioner can be incorporated to improve the conditioning of the coefficient matrix, so that the resulting approach gets rid of the effect of both small and large eigenvalues. The iterative process that results is referred to as a "projection method," and the combined preconditioning is also known as "two-level preconditioning."[48][53]

Simple example of two-level preconditioned system is Conjugate Gradient method (CG)[32][35][51] combined with a two-grid method. In this case, together with the fine-grid linear system from which the approximate solution of the original differential equations is computed, a coarse-grid system is build based on a predefined coarse grid. From a Multi-Grid (MG)[9][24][25] point of view, the (second-level) coarse-grid system is used to reduce the slow-varying, low frequency components of the error, that could not be effectively reduce on the (first-level) fine grid.

Methodology

Schwarz Method with Inflation

Here we try to show that the original Schwarz method need overlapping subdomains in order for convergence. Furthermore, in order to display the pace of the method and however the need of the overlap is connected together. For us to increase the efficiency of the method, we need to introduce what we called the “inflation” and the tool that is needed in creating the overlapping regions by duplicating unknowns at the algebraic level. Once an inflation has

been performed, it leads to the modified Schwarz method. A simple Dirichlet-type interface conditions can be enhanced when we introduce more of complex interface conditions.

Definition 3.1 (**Inflation**).
Let D_Ω represent a domain, that is, a set of unique indices such that each correspond to an unknown. Further let $\{D_{n_I}\}_{I=1\ldots N}$ represent a partition of D_Ω, that is, a set of N disjoint subsets, such that $\bigcup_{I=1}^{N} D_{n_I} = D_n$. Now, each subset D_{n_I} can be inflated into $\tilde{D}_{n_I}$ as follows. Let $i_0, i_1, \ldots i_{N_I}$ denote the indices of D_{Ω_I}. Then $\tilde{D}_{n_I}$ is constructed by adding to D_{n_I} the duplication of the indices j_β belonging to any subset $D_{n_{j\neq I}}$ such that

Inflation

$$\tilde{D}_{n_I} = D_{n_I} \cup \{ j_\beta \in D_{n_I} \mid \exists i_\alpha \in D_{n_I},\ a_{i_\alpha j_\beta} \neq 0 \} \qquad (3.1)$$

Where $a_{i_\alpha j_\beta}$ is the element in the matrix A. The union is the set of inflated subscripts D_Ω^{infl}.

$$\tilde{D}_n = \bigcup_{I=1}^{N} \tilde{D}_{n_I} \quad where\ D_{n_I} < \tilde{D}_{n_I} \qquad (3.2)$$

In the sub-sequence, we will often announce a set of indices which have the same notation than its capitalize identifier, an example is D_{n_I} by I and $\tilde{D}_{n_I}$ also by $\tilde{I}$. Then finally, the inflated operator which is A^{infl} or $\tilde{A}$ actual hold when duplicating the right rows and the right columns.

Modified Schwarz Method

The intense thought of the modified Schwarz method basically consists in using a minimum overlapping which is along with interface conditions improvement, in a way that, more than "Dirichlet data" is actually passed from one subdomain to another in the iterative process. In algebraic terms, these interface condition improvement become an additional sub-block matrices in the inflated matrix and that will be called "interface blocks".

Optimal Algebraic Interface Conditions

When the problem (2.1) is discretized by a finite element or a finite difference method, it yields a linear system $AU = F$. If domain Ω in our problem is decomposed into two subdomains Ω_1 and Ω_2 , at the discrete level this decomposition leads to the matrix partitioning

$$\begin{pmatrix} A_{11} & A_1 & 0 \\ A_1 & A & A_2 \\ 0 & A_2 & A_{22} \end{pmatrix} \begin{pmatrix} U_1 \\ U \\ U_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F \\ F_2 \end{pmatrix}. \qquad (3.3)$$

Where U_Γ corresponds to the unknowns on the interface Γ, and U_j, j = 1, 2 represent the unknowns in the interior of subdomains Ω_1 and Ω_2. In order to write a "modified" (by new interface condition) Schwarz method, we have to introduce two square matrices S_1 and S_2 which act on vectors of the type U_Γ, then the modified Schwarz method reads:

$$\begin{pmatrix} A_{11} & A_1 \\ A_1 & A + S_2 \end{pmatrix} \begin{pmatrix} U_1^{n+1} \\ U_{\Gamma,1}^{n+1} \end{pmatrix} = \begin{pmatrix} F_1 \\ F + S_2 U_{\Gamma,2} - A_2 U_2^n \end{pmatrix} \qquad (3.4a)$$

$$\begin{pmatrix} A_{22} & A_2 \\ A_2 & A + S_1 \end{pmatrix} \begin{pmatrix} U_2^{n+1} \\ U_{\Gamma,2}^{n+1} \end{pmatrix} = \begin{pmatrix} F_2 \\ F + S_1 U_{\Gamma,1} - A_1 U_1^n \end{pmatrix} \qquad (3.4b)$$

Lemma 3.1. Assume that $A + S_1 + S_2$ is invertible and problem in (3.3) is well-posed. Then if the algorithm (3.4) converges, it converges to the solution of (3.3). This is to be understood in the sense that if we denote by $(U_1^{\infty}, U_{,1}^{\infty}, U_2^{\infty}, U_{,2}^{\infty})$ then the limit as n goes to infinity of the sequence
$(U^n, U^n_{,1}, U^n_{,2}, U^n)_{n\geq 2}$ we have for i = 1, 2:
$U_i^{\infty} = U_i$ and $U_{,1}^{\infty} = U_{,2}^{\infty} = U$.

Lemma 3.2. Assume that A_{ii} is invertible for i= 1,2. Then in algorithm (3.4a)-(3.4b), taking

$$S_1 = -A_{1}A_{11}^{-1}A_1$$

and

$$S_2 = -A_{1}A_{22}^{-1}A_2$$

This yields a convergence in two steps.

ADDM Preconditioning

The motivation of our choice of preconditioner is the additive schwarz procedure with its basic form presented in the Algorithm below. Now, it is simple to obtain the preconditioning matrix from the additive schwarz procedure.

Algorithm3.1. Additive Schwarz Iterations.
Required: $D_n = \sum_{i=1}^{N} D_{n_i}$ $thus\ A_i = R_i A R_i^T$

1: **for** $i = 1$ to N **do**
2: Compute $\beta_i = R_i^T A_i R_i (F - AU)$
3: **end for**
4: $U_{new} = U + \sum_{i=1}^{N} \beta_i$

For better picture we need to introduce some notations, that is:

Notation 3.1

$$A_i = R_i A R_i^T \quad partial\ operator\ (i,i) \tag{3.5}$$
$$P_i = R_i^T A^{-1} R_i A \tag{3.6}$$
$$T_i = P_i A^{-1} = R_i^T A^{-1} R_i \tag{3.7}$$

It is notice that, the new iterate in ASM satisfies the relation using the above notation
$U_{new} = (\star - \sum_{i=1} P_i)U + \sum_{i=1}^{N} T_i F$
Thus, this iteration corresponds to a fixed-point iteration $U_{new} = GU + f, with$
$G = I - \sum_{i=1}^{N} P_i,\ f = \sum_{i=1}^{N} T_i F$
With the relation $G = \star - M^{-1}A$ between G and the preconditioning matrix M, the result is that

$$M^{-1}A = \sum_{i=1}^{N} P_i$$

and

$$M^{-1} = \sum_{i=1}^{N} P_i\, A^{-1} = \sum_{i=1}^{N} T_i$$

Now the procedure for applying the preconditioned operator M^{-1} becomes clear (see Algorithm 3.2).
z

Algorithm3.2. Additive Schwarz Preconditioner.
1: Input : $(DDM\ Vector)v$
2: Output: $(DDM\ Vector)z = M^{-1}v$
3: **for** $i = 1$ to N **do**
4: $Compute\ (partial\ vector)z_i := T_i v =$,$endomophic\ partial\ operator(i,i)$-$^{-1}v_i$
5: **end for**
6: Compute $z := z_1 + z_2 + \ldots + z_N$.

Note that, the do loop can be performed in parallel. In line 6 it sums up all the vectors z_i in each domain to obtain global vector z.

Low-Rank Correction

The goal of this low-rank correction is to build a preconditioner for matrix A in the form: $A = \begin{pmatrix} B & E \\ E^T & C \end{pmatrix} = \begin{pmatrix} I & 0 \\ E^T B^{-1} & I \end{pmatrix} \begin{pmatrix} B & E \\ 0 & S \end{pmatrix}$ which is obtained from the DDM.
The preconditioning matrix M is specified in the form of: $M = \begin{pmatrix} I & 0 \\ E^T B^{-1} & I \end{pmatrix} \begin{pmatrix} B & E \\ 0 & S \end{pmatrix}$

Where $S = C - E^T B^{-1} E$, B is diagonals corresponding to the interior unknowns of the decouple subdomains and C is the global interface.
However, we will approximate straightaway the inverse of S instead of S by exploiting low-rank properties. Generally, we seek an approximation of the form:
$\tilde{S}^{-1} = C^{-1} + LRC$, where LRC stands for a low-rank correction matrix. From a practical point of view, it will be difficult to compute directly an approximation in to the matrix$S^{-1} - C^{-1}$, since S^{-1} is not available and we do not (yet) have an efficient means for solving linear systems with the matrix S. Instead we will extract this approximation from that of the matrix A which is defined in the decay properties of $S^{-1} - C^{-1}$. We also consider the expression of $S\ \Xi\ L(I - H)L^T$ and the eigen-decomposition of H in the expression: $H = L^{-1}E^T B^{-1} E L^{-T} = U\text{fl}U^T$,where U is unitary and fl $= diag(\ \lambda_1, \ldots, \lambda_s)$ which is the diagonal matrix of eigenvalues.
This yields:

$$S = L(I - U\text{fl}U^T)L^T = LU(I - \text{fl})U^T L^T \qquad (3.8)$$

The inverse of S is then:

$$S^{-1} = L^{-T}U(I - \text{fl})^{-1}U^T L^{-1},$$

Which is written in the form:

$$S^{-1} = L^{-T}(I + U,(I - \text{fl})^{-1} - I\text{-}U^T)L^{-1},$$
$$S^{-1} = C^{-1} + L^{-T}U,(I - \text{fl})^{-1} - I\text{-}U^T L^{-1}$$

Again, the idea of low-rank approximation came in to mind when a matrix A is considered as symmetric positive definite (SPD). The preconditioner that we want to build or obtained through this approach would be called ADDM preconditioner. The preconditioner is presented in the form:

$$M^{-1} = A_0^{-1} + (A_0^{-1}E)\tilde{G}^{-1}(E^T A_0^{-1}) \quad (3.9)$$

Where $G = I - E^T_\theta A^{-1}E$, now, once we have an approximation of $\tilde{G}^{-1}$ to G^{-1} thus give rise to the application of the ADDM precondition which will have two solves with the A_0.

Implementation of the ADDM Preconditioner

In building the ADDM, we first of all have to consider a graph partitioner which is called on the adjacency graph to partition the domains in the system this is shown in the figure 3-1 below. For each of the subdomain acquired, we then detached both the interior and interface nodes, and then reordering is done on the general matrix. Secondly, we build a solver for each of these $B_{i,a}$ Ξ B_i $+a^{-2}$ $E_i E_i^T$. Note that, the first and the second procedure can be done in parallel. The third step which is the last but not the least, is to build a solver purposely for global matrix such as C_a. Finally, this step is the last and classified as the most expensive stage which is to compute the low-rank approximations:

$$LRC = \tilde{S}^{-1} - C^{-1} \quad (3.10)$$

But,

$$S^{-1} = C^{-1} + L^{-T}U,(I - \text{fl})^{-1} - I\text{-}U^T L^{-1}$$

$$LRC = C^{-1} + L^{-T}U,(I - \text{fl})^{-1} - I\text{-}U^T L^{-1} - C^{-1} \quad (3.11)$$

$$LRC = L^{-T}U,(I - \text{fl})^{-1} - I\text{-}U^T L^{-1} \quad (3.12)$$

We try to apply the ADDM with LR preconditioner by considering the ADDM to be written as:

$$M^{-1} = A_0^{-1}(I + EG_{k,\theta}^{-1} E^T A_0^{-1}) \quad (3.13)$$

The steps involves in the application of M^{-1} to a vector x is shown in the algorithm below. The vector u which is a resultant of the last step is the desire vector $u = M^{-1}x$. To solve the A_o, we need to employed all the five steps in the algorithm.

Algorithm3.3. ADDM Preconditioner with Preconditioning Operations

1: Solve: $A_o z = x$ Solves $\beta_{i,\alpha}$ *and* C_α

2: Compute: $y = E^T Z$

3: Compute: $w = G_{k,\theta}^{-1} y$

4: *Compute*: $v = Ew$

5: **Solve:** $A_o u = x + v$ Solves $\beta_{i,\alpha}$ *and* C_α

Numerical Results

We try to implement the ADDM in Matlab and some numerical results were obtained to compare the performance of ADDM preconditioner to restricted addictive schwarz (RAS) method. The accelerators were conjugate gradient method for matrix and preconditioners which are SPD and generalized minimal residual method for indefinite cases. We then try to compare the preconditioner we have implemented that is ADDM - CG to RAS - GMRES. We also consider a number of factors when comparing the preconditioners after the implementation of the ADDM. We considered the finite difference methods (fdm), number of

subdomains (nDom), number of iterations (n-It), time of iterations (t-It), time of building the preconditioner (t-p) and, finally the residuals.

The results and the analysis of this experiment is shown on experimental result table 4-1 and 4-2.

Results Showing Performance of ADDM without Low Rank Correction

	Fdm	nDom	n-It	t-It	t-p	S=Residuals
32	2	109	114	0.097	1.7779e-09	
4	224	126	1.032	1.6028e-09		
8	320	172	1.531	1.8634e-09		
16	400	205	2.305	8.0138e-09		
64	2	227	237	2.502	7.4740e-10	
4	400	269	3.120	1.8503e-09		
8	400	269	3.120	3.0072e-07		
16	400	269	3.120	5.8094e-05		
128	2	293	307	4.174	7.8237e-10	
4	-	-	-	-		
8	-	-	-	-		
16	-	-	-	-		

From above table 4-1 shows the performance of the ADDM preconditioner without the LRC. We implemented the ADDM in Matlab and the above results were obtained. We loaded the system with a smaller matrix size and keep changing the subdomains in Metis_test python from 2 to the highest subdomain which is 16 as can be shown in table 4-1 above.

Again, we increase the matrix size to a larger system fdm and further increases the size to the largest system of fdm to obtained the results.

From the table 4-1, it can be seen that, as the number of subdomains increases, the number of iteration count also increases alongside and whiles the time of iteration increases, the time in building the preconditioner also increases and thereby minimizing the residuals in both fdm 32 and 64 respectively.

For the case of fdm 128, we only have the iteration count for only for the smallest subdomain and the rest of the subdomains failed to performed. We can however conclude that, our choice of preconditioner is a better preconditioner since it allows for convergence to be achieved within a very possible fewest iterations with the best iteration iteration time.

Results Showing Performance of ADDM with Low Rank Correction

fdm	nDom	rk	n-It	t-It	t-p	S=Residuals
32	2	4	115	122	1.022	1.8351e-07
	4	8	236	130	1.731	1.8002e-07
	8	16	342	193	2.036	1.8721e-07
	16	32	400	237	2.825	8.3570e-07
64	2	4	246	257	3.163	8.7822e-09
	4	8	280	293	3.902	2.9821e-05
	8	16	322	331	4.011	3.6443e-05
	16	32	400	363	4.302	6.6026e-06
128	2	4	306	342	4.402	8.9114e-09
	4	8	315	349	4.653	7.5464e-09
	8	16	337	363	4.880	7.9992e-09
	16	32	352	386	5.127	7.1251e-09

From above table 4-2 shows the performance of the ADDM preconditioner with the LRC.
From the performance with low rank correction, it can be shown that, there have been a significant improvement in the number of iterations, the time of iterations and the time to build the preconditioner as a result of the increase in the ranks of the matrix. The main issue with the preconditioner is the fact that it is difficult to update. For instance, when we computed the preconditioner to the first matrix, we finds out that it was not accurate enough in order to yield convergence. In the case of the ADDM we started all over from the beginning instead of updating it from where the error is.
However, after the introduction of the low rank, the preconditioner is improved by obtaining a low rank correction matrix *G*, where $G = I - E^T A_0^{-1} E$. And adding a few more vectors that is increase k and this can be achieved in a number of ways without having to throw away the vectors already computed which can even consume more time.

Conclusions

Algebraic domain decomposition methods have been our main focused in this thesis, and by that, we were to designed a scalable ADDM preconditioner in handling large linear systems or problems. We were able to have access to the coefficients of the matrix for the linear systems to be solved. We also have introduced some two fresh algebraic method for building interface conditions. The most important fact about the methods are that, both methods are adaptive and can also be used during the first solve, that is, even before the first solve is completed.

For both methods, an information is pull out from the Krylov space produce by a few iterations of the Schwarz method algorithm with an overlapping subdomains. We also consider the Ritz eigenvectors which corresponds to the low eigenvalues since they are accountable for the inactivity or the slowness of the Krylov solver preconditioned by the Schwarz algorithm. A modification of another method arises from the changes in the algorithm of ASM to MSM. We were able to also check for their convergence level by the influence of the inflation in the system.

References

[1] Multi-Core Strategies[C]: Mpi And Open Mp. Http://Www.Hpccommunity. Org/f55/ Multi-Core-Strategies-Mpi-Openmpi-702/.

[2] Achdou, Y., And Nataf, F. A Robin-Robin Preconditioner For An Advection-Diffusion Problem[J]. C. R. Acad. Sci. Paris 325, Série I (1997), 1211–1216.

[3] Achdou, Y., Tallec, P. L., Nataf, F., And Vidrascu, M. A Domain Decomposition Preconditioner For An Advection-Diffusion Problem[J]. Comp. Meth. Appl. Mech. Engrg 184 (2000), 145–170.

[4] Aldous, J., And Wilson, R. J. Graphs And Applications: An Introductory Approach[M]. Springer London Ltd., 2003.

[5] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., Mckenney, A., And Sorensen, D. Lapack Users' Guide, Third Ed. Society For Industrial And Applied Mathematics[M], Philadelphia, Pa, 1999.

[6] Balay, S., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., Mcinnes, L. C., Smith, B. F., And Zhang, H. Petsc Users Manual[J]. Tech. Rep. Anl-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[7] Balay, S., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., Mcinnes, L. C., Smith, B. F., And Zhang, H. Petsc Web Page[C], 2009. Http://Www.Mcs.Anl.Gov/Petsc.

[8] Balay, S., Gropp, W. D., Mcinnes, L. C., And Smith, B. F. Efficient Management Of Parallelism In Object Oriented Numerical Software Libraries[J]. In Modern Software Tools In Scientific Computing (1997), E. Arge, A. M. Bruaset, And H. P. Langtangen, Eds., Birkhäuser Press, Pp. 163–202.

[9] Bastian, P., Hackbush, W., And Wittum, G. Additive And Multiplicative Multi-Grid - a Comparison[J]. Computing 60 (1998), 345–346.

[10] Broquedis, F., Clet Ortega, J., Moreaud, S., Furmento, N., Goglin, B., Mercier, G., Thibault, S., And Namyst, R. Hwloc: a Generic Framework For Managing Hardware Affinities In Hpc Applications. In Pdp 2010 - The 18th Euromicro International Conference On Parallel, Distributed And Network-Based Computing[C] (Pisa Italie, 02 2010), Ieee, Ed.

[11] Buluç, A., Fineman, J. T., Frigo, M., Gilbert, J. R., And Leiserson, C. E. Parallel Sparse Matrix-Vector And Matrix-Transpose-Vector Multiplication Using Compressed Sparse Blocks. In Spaa '09: Proceedings Of The Twenty-First Annual Symposium On Parallelism In Algorithms And Architectures [J](New York, Ny, Usa, 2009), Acm, Pp. 233–244.

[12] Cai, X.-C., And Sarkis, M. A Restricted Additive Schwarz Preconditioner For General

Sparse Linear Systems[J]. Siam Journal On Scientific Computing 21 (1999), 239–247.

[13] Chan, T., Glowinski, R., Périaux, J., And Widlund, O., Eds. Domain Decomposition Methods (Philadelphia, Pa, 1989), Siam. Proceedings Of The Second International Symposium On Domain Decomposition Methods[C], Los Angeles, California, January 14 - 16, 1988.

[14] Chan, T. F., And Mathew, T. P. Acta Numerica. Cambridge University Press, 1994, Ch. Domain Decomposition Algorithms[J].

[15] Chen, Z., Huan, G., And Ma, Y. Computational Methods For Multi-phase Flows In Porous Media. Society For Industrial And Applied Mathematics[J], Dallas, Texas, 2006.

[16] Chevalier, C., And Pellegrini, F. Pt-Scotch: a Tool For Efficient Parallel Graph Ordering[J]. Parallel Computing 6-8, 34 (2008), 318–331.

[17] Collino, F., Ghanemi, S., And Joly, P. Domain Decomposition Methods For Harmonic Wave Propagation[C]: a General Presentation. Comput. Methods Appl. Mech. Engrg, 2-4 (2000), 171–211.

[18] Demmel, J. W., Eisenstat, S. C., Gilbert, J. R., Li, X. S., And Liu, J. W. H. A Super-Nodal Approach To Sparse Partial Pivoting[J]. Siam J. Matrix Analysis And Application 20, 3 (1999), 720–755.

[19] Després, B. Domain Decomposition Method And The Helmholtz Problem.Ii. In Second International Conference On Mathematical And Numerical Aspects Of Wave Propagation[C] (Newark, De, 1993) (Philadelphia, Pa, 1993), Siam, Pp. 197–206.

[20] Dryja, M., And Widlund, O. B. Towards a Unified Theory Of Domain Decomposition Algorithms For Elliptic Problems[C]. In Third International Symposium On Domain Decomposition Methods For Partial Differential Equations (1990), T. Chan, R. Glowinski, J. Périaux, And O. B. Widlund, Eds., Siam, Philadelphia, Pa, Pp. 3–21.

[21] Durlofsky, L. J. A Triangle Based Mixed Finite Element-Finite Volume Technique For Modeling Two Phase Low Through Porous Media[J]. J. Comput. Phys., 105 (1993), 252–266.

[22] Efstathiou, E., And Gander, M. J. Why Restricted Additive Schwarz Converges Faster Than Additive Schwarz[180]. Bit Numerical Mathematics 43 (2003), 945–959.

[23] Erlangga, Y. A., And Nabben, R. Deflation And Balancing Preconditioners For Krylov Subspace Methods Applied To Non-Symmetric Matrices[J]. Siam J. Matrix Anal. Appl. 30 (2008), 684–699.

[24] Flauraud, E., Nataf, F., And Willien, F. Optimized Interface Conditions For Domain Decomposition Methods For Problems With Extreme Contrast In The Coefficients. Journal Of Computational And Applied Mathematics[J] 189 (2006), 539–554.

[25] Gamma, E. Design Patterns: Elements Of Reusable Object-Oriented Software. Addison Wesley, 2004.

[26] Gander, M. J. Schwarz Methods In The Course Of Time. Etna 31 (2008), 228–255.

[27] Gander, M. J., Halpern, L., Magoulès, F., And Roux, F. X. Analysis Of Patch Sub-Structuring Methods[9]. Int. J. Appl. Math. Comput. Sci. 17, 2 (2007), 395–402.

[28] Gander, M. J., Magoulès, F., And Nataf, F. Optimized Schwarz Methods Without Overlap For The Helmholtz Equation. Siam J. Sci. Comput. 24, 1 (2002), 38–60.

[29] Goossens, S., And Roose, D. Ritz And Harmonic Ritz Values And The Convergence Of Fom And Gmres. Numerical Linear Algebra With Applications [M]6, 4 (Sep 1999),

281–293.

[30] Gosselet, P., And Rey, C. On a Selective Reuse Of Krylov Sub-Spaces In Newton - Krylov Approaches For Nonlinear Elasticity. In Proceedings Of The Fourteenth International Conference On Domain Decomposition Methods Fourteenth International Conference On Domain Decomposition Methods[C] (Cocoyoc Mexique, 2003), O. W. I. Herrera, D. Keyes And R. Yates, Eds., Pp. 419–426.

[31] Greenbaum, A. Iterative Methods For Solving Linear Systems. Siam, 1997.

[32] Grop, W., Lusk, E., And Skjellum, A. Using Mpi: Portable Parallel Programming With Message-Passing Interface. Mit Press, 1994.

[33] Hagstrom, T., Tewarson, R. P., And Jazcilevich, A. Numerical Experiments On a Domain Decomposition Algorithm For Nonlinear Elliptic Boundary Value Problems. Appl. Math[J]. Lett. 1, 3 (1988).

[34] Hecht, F. Freefem++, 3.7 Ed. Numerical Mathematics And Scientific Computation[M]. Laboratoire J.L. Lions, Université Pierre Et Marie Curie, Http://Www.Freefem.Org/Ff++/, 2010.

[35] Hestenes, M., And Stiefel, E. Methods Of Conjugate Gradient For Solving Linear Systems[J]. J. Res. Nat. Bur. Stand. 49 (1952), 409–436.

[36] Karypis, G., And Kumar, V. A Fast And Highly Quality Multilevel Scheme For Partitioning Irregular Graphs. Siam Journal On Scientific Computing 20, 1 (1999), 359–392.

[37] Lai, C.-H., Bjørstad, P. E., Cross, M., And Widlund, O., Eds. Eleventh International Conference On Domain Decomposition Methods (1998). Proceedings Of The 11th International Conference On Domain Decomposition Methods[C] In Greenwich, England, July 20-24, 1998.

[38] Le Tallec, P. Domain Decomposition Methods In Computational Mechanics[M]. In Computational Mechanics Advances, J. T. Oden, Ed., Vol. 1 (2). North-Holland, 1994, Pp. 121– 220.

[39] Lions, P.-L. On The Schwarz Alternating Method. Iii: a Variant For Non-Overlapping Subdomains. In Third International Symposium On Domain Decomposition Methods For Partial Differential Equations[C], Held In Houston, Texas, March 20-22, 1989 (Philadelphia, Pa, 1990), T. F. Chan, R. Glowinski, J. Périaux, And O. Widlund, Eds., Siam.

[40] Magoulès, F., Roux, F.-X., And Salmon, S. Optimal Discrete Transmission Condition For Non-Overlapping Domain Decomposition Method For Helmholtz Equation[J]. Siam Journal On Scientific Computing 25, 5 (2004), 1947–1515.

[41] Magoulès, F., Roux, F. X., And Series, L. Algebraic Approximation Of Dirichlet-To-Neumann Maps For Equations Of Linear Elasticity[J]. Comp. Meth. Appl. Mech. Engrg. 195 (2006), 3742–3759.

[42] Mandel, J. Balancing Domain Decomposition. Communications In Applied And Numerical Methods[J] 9 (1993), 233–241.

[43] Matsokin, A. M., And Nepomnyaschikh, S. V. A Schwarz Alternating Method In a Subspace[J]. Soviet Mathematics 29(10) (1985), 78–84.

[44] Nataf, F., And Rogier, F. Factorization Of The Convection-Diffusion Operator And The Schwarz Algorithm[J]. M3As 5, 1 (1995), 67–93.

[45] Nataf, F., Rogier, F., And De Sturler, E. Optimal Interface Conditions For Domain Decomposition Methods[J]. Tech. Rep. 301, Cmap (Ecole Polytechnique), 1994.

[46] Nataf, F., Xiang, H., And Dolean, V. A Coarse Space Construction Based On Local Dtn Maps[C]. Http://Hal.Archives-Ouvertes.Fr/Hal-00491919/Fr/, 2010.

[47] Nicolaides, R. A. Deflation Of Conjugate Gradients With Applications To Boundary Value Problems[J]. Siam J. Matrix Anal. Appl. 24 (1987), 355–365.

[48] Padiy, A., Axelsson, O., And Polman, B. Generalized Augmented Matrix Preconditioning Approach And Its Application To Iterative Solution Of Ill-Conditioned Algebraic Systems[J]. Siam J. Matrix Anal. Appl. 22 (2000), 793–818.

[49] Petter E. Bjørstad, M. S. E., And Keyes, D. E., Eds. 9th International Conference On Domain Decomposition Methods, held in Bergen, Norway (1997)[C].

[50] Quarteroni, A., And Valli, A. Domain Decomposition Methods For Partial Differential Equations[D]. Oxford Science. Publication, Oxford University Press, (1999).

[51] Saad, Y. Iterative Methods For Sparse Linear Systems[M], 2nd Edition Siam, Philadelphia, 2003.

[52] Saad, Y., And Schultz, M. H. A Generalized Minimal Residual Algorithm For Solving Non-Symmetric Linear Systems[J]. Siam J. Science. Stat. Comput. 7, 3 (1986).

[53] chwarz, H. A. Über Einen Grenzübergang Durch Alternierendes Verfahren[J]. Vierteljahrsschrift Der Naturforschenden Gesellschaft In Zürich 15 (May 1870), 272–286.