Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 2580820, 11 pages
https://doi.org/10.1155/2017/2580820

Research Article

Parallel Multiprojection Preconditioned Methods Based on Subspace Compression

Byron E. Moutafis, Christos K. Filelis-Papadopoulos, and George A. Gravvanis

Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, University Campus, Kimmeria, 67100 Xanthi, Greece

Correspondence should be addressed to Christos K. Filelis-Papadopoulos; cpapad@ee.duth.gr

Received 1 February 2017; Revised 5 April 2017; Accepted 11 April 2017; Published 30 July 2017

Academic Editor: Damijan Markovic

During the last decades, the continuous expansion of supercomputing infrastructures necessitates the design of scalable and robust parallel numerical methods for solving large sparse linear systems. A new approach for the additive projection parallel preconditioned iterative method based on semiaggregation and a subspace compression technique, for general sparse linear systems, is presented. The subspace compression technique utilizes a subdomain adjacency matrix and breadth first search to discover and aggregate subdomains to limit the average size of the local linear systems, resulting in reduced memory requirements. The depth of aggregation is controlled by a user defined parameter. The local coefficient matrices use the aggregates computed during the formation of the subdomain adjacency matrix in order to avoid recomputation and improve performance. Moreover, the rows and columns corresponding to the newly formed aggregates are ordered last to further reduce fill-in during the factorization of the local coefficient matrices. Furthermore, the method is based on nonoverlapping domain decomposition in conjunction with algebraic graph partitioning techniques for separating the subdomains. Finally, the applicability and implementation issues are discussed and numerical results along with comparative results are presented.

1. Introduction

Let us consider a sparse linear system of the following form:

$$Ax = b, \tag{1}$$

where A is the coefficient matrix, b is the right-hand-side vector, and x is the solution vector. In order to solve the linear system, in (1), a direct or an iterative method can be used. However, a direct method is computationally expensive with excessive memory requirements. Krylov subspace iterative methods, compare [1], have gained the attention of the scientific community during the recent decades, due to their efficiency and memory requirements for solving large sparse linear systems. Preconditioned iterative methods improve the convergence rate and have been used extensively for solving large sparse linear systems. The left preconditioned form of the linear system, in (1), is as follows:

$$M^{-1}Ax = M^{-1}b, \tag{2}$$

where M^{-1} is a suitable preconditioning matrix. Moreover, there is the right and the symmetric preconditioned form, compare [1]. The preconditioner M^{-1} has to satisfy the following conditions: (i) $M^{-1}A$ should have a "clustered" spectrum, (ii) M^{-1} can be efficiently computed in parallel, and (iii) the product of "M^{-1} by vector" should be fast to compute in parallel, compare [1, 2]. The domain decomposition method has been extensively used for solving (very) large sparse linear systems in parallel systems, compare [3–8]. Furthermore, domain decomposition-type methods have been combined with Krylov subspace methods or have been used in hybrid schemes in conjunction with direct solvers, compare [6, 9–11]. During the last decade, research efforts were directed towards solvers for general sparse linear systems that have improved convergence behavior and are scalable.

Various overlapping domain decomposition methods based on Lagrange multipliers have been proposed. The Finite Element Tearing and Interconnect (FETI), compare [12], and its variants, compare [13, 14], are scalable and efficient

solvers for second-order and fourth-order partial differential equation problems. Related methods are the Balancing Domain Decomposition (BDD), compare [15], and the Balancing Domain Decomposition by constraints (BDDC), compare [16], which are multiplicative and additive domain decomposition methods, respectively. The aforementioned methods are suitable for solving linear systems derived from the finite element method. Additionally, there are domain decomposition-type solvers using graph partitioning algorithms. These algebraic domain decomposition methods do not require any geometric property of the problem as the partitioning is based on the corresponding graph of the coefficient matrix. Parallel hybrid algebraic domain decomposition-type solvers that combine direct and iterative methods have been proposed, compare [17, 18]. Additionally, algebraic domain decomposition method in conjunction with algebraic multigrid method, compare [19–22], has satisfactory convergence behavior and is suitable for parallel systems. An algebraic domain decomposition method, based on semiaggregation and subspace compression techniques, namely, multiprojection method with subspace compression (MPMSC), for solving large sparse linear systems is proposed. Several efficient software libraries for graph partitioning, such as Metis, compare [23], or Scotch, compare [24], can be used for algebraic partitioning. The aggregation techniques have been previously used in the context of algebraic multigrid methods and have been shown to be efficient for a wide variety of model problems, compare [25–27]. In the MPMSC scheme, a direct method is used for solving the local linear systems derived from the semiaggregated domains. The semiaggregation procedure results in multiple grids composed of a fine part, corresponding to the unknowns of a subdomain, and aggregated part, corresponding to unknowns related to other subdomains. The proposed scheme utilizes approximations for the fine part of each subdomain, obtained with respect to its aggregated components, as opposed to aggregation based algebraic multigrid methods, which obtain approximations for the solution of the entire domain, through smoothing, based on a hierarchy of coarser grids. Moreover, the proposed scheme does not require the computation of a hierarchy of coarser grids. The subspace compression technique is an algorithmic procedure, based on breadth first search with limited depth on the adjacency matrix of the subdomains. By this technique the neighboring subdomains are discovered and aggregated together to an aggregated (coarse) component. The aggregation process reduces the number of aggregated (coarse) unknowns of the local linear systems, resulting in a substantial reduction in memory requirements. The computation of the rows and columns corresponding to the aggregates, during the formation of the local coefficient matrices as well as the subdomain adjacency matrix, is reused to avoid recomputation. Avoiding the recomputation enhances the performance of the proposed scheme. Moreover, the rows and columns corresponding to the newly formed aggregates have a large number of nonzero elements and thus are ordered last to further reduce fill-in during the factorization of the local coefficient matrices. Further, the number of aggregates of the multiprojection method with subspace compression is controlled by a parameter that describes the level sets of aggregates to be compressed in a single aggregate.

In Section 2, the multiprojection method with subspace compression is presented, implementation details are given, and the aggregation strategy is discussed. In Section 3, numerical results illustrating the applicability and efficiency of the MPMSC scheme are presented. Moreover, comparative results for the performance and convergence behavior are given.

2. Multiprojection Method with Subspace Compression

The multiprojection method with subspace compression is a domain decomposition-type method for solving large sparse linear systems (see (1)). The graph, corresponding to the coefficient matrix A, has vertices, the unknowns of the linear system, in (1), and edges, the nonzero elements of the coefficient matrix A. Since A is expressed as a graph, the algebraic properties of the problem can be utilized in order to partition the graph into n_{doms} nonoverlapping subgraphs (subdomains). The graph partitioning process is performed using the algorithm provided in the Metis software library, compare [23]. Consider the domain Ω partitioned into n_{doms} nonoverlapping subdomains Ω_j with $j = 0, \ldots, n_{\text{doms}} - 1$, so that

$$\Omega = \bigcup_{j=0}^{n_{\text{doms}}-1} \Omega_j. \tag{3}$$

The number of vertices, in each subdomain Ω_j, is denoted by m_j and S_j is the index set defined by the following expression:

$$S_j = \left\{ i_1, i_2, \ldots, i_{m_j} \right\}, \tag{4}$$

where i_k are the indices corresponding to the m_j vertices of each subdomain.

2.1. Subspace Compression. Let us consider the subspace K, of the domain Ω, which contains n_{doms} aggregated (coarse) components, corresponding to the subdomains Ω_j, and the prolongation matrix R from K onto Ω, $R : \mathrm{K} \to \Omega$, which is denoted according to the following expression:

$$R_{i,j} = \begin{cases} \dfrac{1}{m_j}, & \text{if } i \in S_j,\ i = 0, \ldots, n-1, \\ 0, & \text{if } i \notin S_j,\ j = 0, \ldots, n_{\text{doms}} - 1. \end{cases} \tag{5}$$

The projection of A to subspace K is the matrix H of dimension $(n_{\text{doms}} \times n_{\text{doms}})$ and is given by the following equation:

$$H = R^T A R. \tag{6}$$

The graph $G(H) = (v, e)$, corresponding to the matrix H, concerns the relations between the aggregated (coarse) components. The subspace compression technique is based on the reaggregation of the n_{doms} (number of subdomains) aggregated (coarse) components into n_{aggs} (number of aggregates).

The number of aggregates is not known a priori and thus is a result of the subspace compression algorithm. Breadth first search with limited depth (BFS-LD) is performed on $G(H)$ to find the distance d neighborhood. The components in a distance d neighborhood of $G(H)$ are reaggregated together. Then the neighborhood's vertices are removed from the set v and the BFS-LD is performed again on the new graph. When every vertex is removed from the set v, the procedure is terminated. The order in which the reaggregation process is performed may result in a slight variation of the number of aggregates; however this does not affect the efficiency or the convergence behavior of the proposed scheme significantly. The order of the vertices used in the reaggregation scheme is lexicographical with respect to the enumeration of the subdomains by the graph partitioning scheme. Other approaches can be used for the reaggregation process, compare [26, 27]; however the computational work of these schemes will affect negatively the performance of the preprocessing phase, especially for large numbers of subdomains. The subspace compression algorithm is as follows:

(1) $U \leftarrow K$, $\text{Agg} \leftarrow \{\text{Agg}_0, \text{Agg}_1, \ldots, \text{Agg}_{n_{\text{aggs}}-1}\} = \{\emptyset, \emptyset, \ldots, \emptyset\}$.

(2) **For each** $y_i \in U$ or **until** $U \supset \emptyset$

$\text{Agg}_k \leftarrow N_{y_i}^d \cap U$, where $N_{y_i}^d$ is the d-distance neighborhood of component y_i and k is the index of the aggregated (coarse) components

$$U \leftarrow U \setminus \text{Agg}_k \tag{7}$$

End For

The parameter d of the neighborhood $N_{y_i}^d$ specifies the maximum distance between the vertex y_i and any other vertex in the neighborhood. The value of d is selected with respect to the adjacency matrix $G(H)$, considering that the larger the value of d, the less the number of aggregates. Distance one neighborhood is chosen by default; however for parallel systems with limited memory resources, distance two or distance three neighborhood should be chosen to reduce the memory requirements of the proposed preconditioning scheme. An increase in the value of d might affect negatively the convergence behavior of the proposed scheme. The computational complexity for finding the aggregates depends on the number of subdomains and it is a searching procedure on the graph $G(H)$ with negligible execution time compared to the whole initialization phase of the preconditioning scheme.

2.2. Semiaggregation. Let us consider the semiaggregated subdomains Z_j, containing the m_j fine components of subdomain Ω_j and n_{aggs} aggregated components. Furthermore, the semiaggregated Z_j subdomains are associated with their respective prolongation matrices $V_j \in \mathbb{R}^{n\times(m_j+(n_{\text{aggs}}))} : Z_j \rightarrow \Omega$, with $j = 0, 1, \ldots, n_{\text{doms}} - 1$. The matrices $V_j^T : \Omega \rightarrow Z_j$ are restriction operators mapping an arbitrary full vector of the domain Ω into a vector of the semiaggregated subdomain Z_j with size $(m_j + n_{\text{aggs}})$. The first m_j columns of the matrix V_j are the columns of the identity matrix with dimension $(n\times n)$, corresponding the indices S_j. The remaining (n_{aggs}) column vectors of V_j correspond to the aggregates. The components of the column vector that are included in the corresponding aggregated subdomain have value $1/M_k$ and the rest of the components are zero, where $M_k = \sum m_i[S_i \subseteq \text{Agg}_k]$, with $i \neq j$ and $k = 0, 1, \ldots, n_{\text{aggs}} - 1$.

Hence, the matrix V_j is defined as follows:

$$V_j = \left[e_k,\ k \in S_j \mid q_k,\ 0 \leq k < n_{\text{aggs}}\right], \tag{8}$$

where q_k is the column vectors corresponding to the aggregates as they are described above. Moreover, additional prolongation matrices are required, namely, W_j. The prolongation matrices W_j of dimension $(n \times (m_j + n_{\text{aggs}}))$ are used to map a vector defined on the subdomain Z_j to a vector defined on the domain Ω by prolonging only the components corresponding to the subdomain Ω_j and discarding the aggregated (coarse) components. The ith column of the matrix W_j is given as follows:

$$\left(W_j\right)_{:,i} = \begin{cases} e_i, & \text{if } i \in S_j, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The linear systems corresponding to each subdomain Z_j are as follows:

$$A_j x_j = V_j^T b = b_j, \tag{10}$$

where $A_j = V_j^T A V_j$ is a coefficient matrix of order $(m_j + (n_{\text{aggs}}))$.

The proposed method is used as a preconditioner to a Krylov subspace iterative method, such as preconditioned GMRES(m), compare [28, 29]. The product of the preconditioning matrix by a vector r can be described by the following compact algorithmic scheme:

For $j = 0, \ldots, n_{\text{doms}} - 1$

$$\text{Compute } y_j = W_j W_j^T V_j A_j^{-1} V_j^T r \tag{11}$$

End For

$$y = \sum_{j=0}^{n_{\text{doms}}-1} y_j. \tag{12}$$

Considering that X_* is the exact solution and that $r = b - Ax = A(x_* - x) = Ae$, the error e_{k+1} at $(k+1)$th iteration can be written as follows:

$$e_{k+1} = e_k - \sum_{j=0}^{n_{\text{doms}}-1} W_j W_j^T V_j A_j^{-1} V_j^T A e_k, \tag{13}$$

or equivalently

$$e_{k+1} = \left(I - \sum_{j=0}^{n_{\text{doms}}-1} Q_j\right) e_k, \tag{14}$$

where

$$Q_j = W_j W_j^T V_j A_j^{-1} V_j^T A. \tag{15}$$

$W_j W_j^T$ is an orthogonal projector with respect to the Euclidean inner product. Considering that if A is symmetric positive definite, $V_j A_j^{-1} V_j^T A$ is an orthogonal projector with respect to the A-norm inner product, compare [8, 30]. Q_j is a product of the two orthogonal projections, which are orthogonal to different inner products; thus the projection is not orthogonal, compare [31]. Taking into consideration the above, the method is categorized into the class of oblique projection procedures, compare [1].

The preconditioned matrix using the MPMSC scheme is given as follows:

$$M_{\text{MPMSC}} A = \sum_{j=0}^{n_{\text{doms}}-1} Q_j, \tag{16}$$

which is a sum of projections.

Let us consider the unit square domain Ω, which is discretized with mesh size $h = 1/7$ as depicted in Figure 1. The resulting grid is partitioned into 16 subdomains, containing 4 vertices each. The corresponding matrix A is projected onto the subdomains Z_j, in (10), and the grid corresponding to the projected subdomain Z_0 is presented schematically in Figure 2.

In the MPMSC scheme, n_{doms} $((m_j + n_{\text{aggs}}) \times (m_j + n_{\text{aggs}}))$ linear systems of the form in (10) have to be solved in every iteration. Equivalently, the linear system, in (10), can be written in a reordered block form as follows:

$$\begin{bmatrix} B_{FF} & B_{FC} \\ B_{CF} & B_{CC} \end{bmatrix} \begin{bmatrix} x_F \\ x_C \end{bmatrix} = \begin{bmatrix} b_F \\ b_C \end{bmatrix}, \tag{17}$$

where the size of B_{FF} is $(m_j \times m_j)$ and the size of B_{CC} is $((n_{\text{aggs}}) \times (n_{\text{aggs}}))$. The vector x_F is prolonged to the full solution vector x of domain Ω, and the aggregated (coarse) components are used as auxiliary. In the MPMSC preprocessing step each matrix A_j is factorized by sparse LU factorization method, compare [32]:

$$A_j = L_j U_j. \tag{18}$$

Therefore, the preconditioning operation can be described by the following algorithmic scheme:

For $j = 0, \ldots, n_{\text{doms}} - 1$

$$\text{Solve the } L_j w_j = V_j^T x \tag{19}$$

$$\text{Solve the } U_j z_j = w_j \tag{20}$$

$$\text{Compute } y_j = W_j W_j^T V_j z_j \tag{21}$$

End For

$$y = \sum_{j=0}^{n_{\text{doms}}-1} y_j. \tag{22}$$

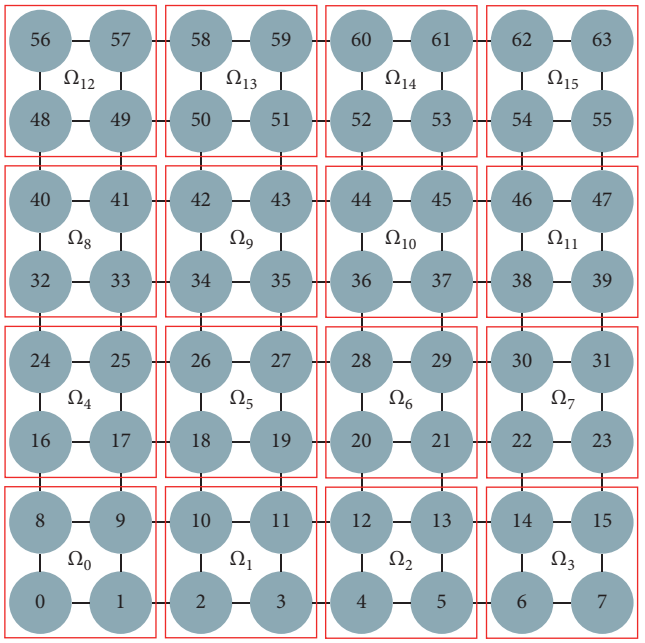

Figure 1: A square domain Ω discretized with mesh size $h = 1/7$ partitioned into 16 subdomains Ω_j.

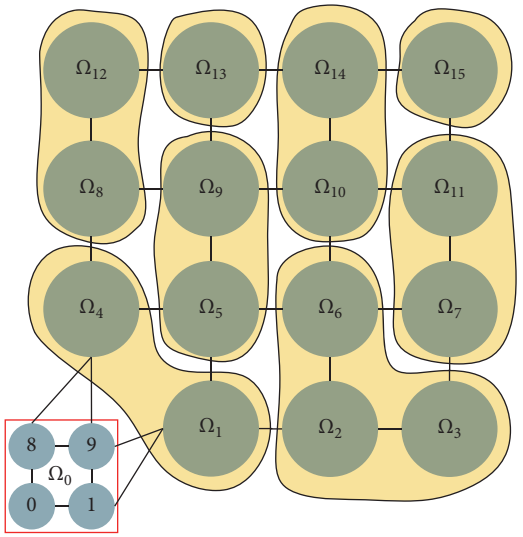

Figure 2: The grid corresponding to the projection of the domain Ω onto subdomain Z_0.

The solution, in (19)-(20), of the local linear systems is performed inherently in parallel; thus each core computes one backward-forward substitution process. The aggregated (coarse) components are ordered last in the coefficient matrix A_j, because they have many connections with the fine components and consequently many nonzeros in their rows/columns. By this ordering, the fill-in of the L and U matrices is reduced, compare [32]. Direct solvers have memory and computational complexity limitations for large linear systems. In order to keep the size of A_j below these limits, the coefficient matrix A should be partitioned into a

suitable number of subdomains. Maintaining the size of A_j almost constant, regardless of the size of A, results in almost constant LU factorization computational work.

It should be stated that the MPMSC scheme is inherently parallel and thus suitable for distributed systems. The MPMSC scheme is used as a preconditioner to the Parallel Preconditioned restarted Generalized Minimum RESidual, namely, (PPGMRES(m)), method which is a parallel variant of the preconditioned GMRES(m) method proposed by Douglas et al., compare [28], and is based on Gram–Schmidt orthogonalization, compare [33].

2.3. Memory Requirements. The memory requirements of a sparse matrix of size n with nnz nonzero elements stored in CSR (Compressed Sparse Row) format are denoted by

$$\begin{aligned} \text{csr}(n, \text{nnz}) = ((n+1) * \text{sizeof}(\text{int}) + \text{nnz} \\ * \text{sizeof}(\text{int}) + \text{nnz} * \text{sizeof}(\text{double})). \end{aligned} \tag{23}$$

The size of the coefficient matrix A is denoted by "n", $E[]$ is the expected value operator, and nnz() is an operator returning the number of nonzero elements of a matrix. Also, let us denote by "n_{node}" the average number of components that correspond to a node, which usually is equal to (cores per node) $*$ $E[m_j]$. The restart parameter of PPGMRES(m) is denoted by "m." MPMSC requires the following matrices and vectors in the preprocessing phase.

The coefficient matrix A in CSR with memory requirements csr(n, nnz(A)), the right-hand-side vector b ($n*$sizeof(double)), the partitioning vector ($n*$sizeof(int)) and (number of subdomains/nodes) vectors that map the local index of the components of each subdomain to the global index ((number of subdomains/nodes) $*$ $E[m_j]$ $*$ sizeof(int)), and also the (number of subdomains/nodes) sparse (CSR) matrices V_j with memory requirements csr(n, n) are required. The matrices W_j are not stored, since they can be applied explicitly. Moreover, the LU factorization process requires a large fraction of the total memory requirements. Each node requires (number of subdomains/nodes) L_j and U_j sparse (CSR) matrices with memory requirements csr($(m_j + n_{\text{aggs}} + 1), E[\text{nnz}(L_j)]$) and csr($(m_j + n_{\text{aggs}} + 1), E[\text{nnz}(U_j)]$). The MPMSC requires also the sparse (CSR) matrices H (csr(n_{doms}, nnz(H))) and R (csr(n, n)) for the subspace compression process; however they are temporary as they are deleted before the LU factorization step. The PPGMRES(m) algorithm has the following extra memory requirements: six auxiliary vectors ($n_{\text{node}} *$ sizeof(double)), the dense matrix retaining the basis vectors of the Krylov subspace ($n*(m+1)*$sizeof(double)), the upper Hessenberg matrix T ($(m+1)*m*$sizeof(double)), and three auxiliary vectors for the Givens rotation ($(m+1)*$sizeof(double)).

3. Numerical Results

In this section, the effectiveness and applicability of the MPMSC scheme are examined. The numerical experiments were performed on a BlueGene/P (BG/P) supercomputer with the following specifications: CPU: 1024x Quad-Core PowerPC-450 850 Mhz; RAM: 4 GB/node; interconnect: 3D-Torus network with bandwidth 5.1 GBps. The methods are designed for distributed systems with multicore nodes, using MPI and OpenMP environments, compare [34]. Each multicore node executes an MPI process, which undertakes the computations for as many subdomains as the number of its cores. The parallelization in each multicore node is achieved with OpenMP, so that every subdomain is mapped to one core. Having equal number of cores and subdomains allows the parallel computation of every LU factorization. Moreover, the software libraries BLAS (Basic Linear Algebra Subprograms), compare [35], and Metis, compare [23], have been used. The restart parameter of PPGMRES(m) was set to 20 and the maximum number of iterations was set to 500 iterations. The termination criterion was $\|r\|_2 < 10^{-8}\|b\|_2$. The execution time is given in "seconds." The number of nonzero elements in the matrix A is denoted by nnz(A). The convergence behavior of the PPGMRES(m) is given by outer (inner) iterations. The inner iterations are the number of iterations after the last restart of the PPGMRES(m) and outer iterations are the number of times PPGMRES(m) was restarted. The neighborhood distance parameter d has been set to one, unless stated otherwise explicitly.

The formation of the prolongation operators and the LU factorization of the local matrices are included in the initialization phase of the preconditioning scheme, which is part of the preprocessing time (Pre-Time). The time required for the graph partitioning, using Metis, compare [23], is not included in the Pre-Time. It should be noted that parallel graph partitioning schemes, such as ParMETIS, compare [36], can be used to increase parallel performance. The communication time of the PPGMRES(m) is given as Comm-Time and the GMRES-Time is the execution time of the PPGMRES(m). It should be mentioned that the Comm-Time is included in the GMRES-Time. Total-Time is the sum of the Pre-Time and the GMRES-Time. The speedup is computed using the execution on 64 cores as a baseline unless stated otherwise explicitly.

3.1. 3D Poisson Problem. The PPGMRES(m) in conjunction with MPMSC scheme has been used for solving the Poisson equation in three space variables subject to Dirichlet boundary conditions. The Poisson equation was discretized with the 7-point stencil. The right-hand side of the linear systems was computed as the product of coefficient matrix A and the solution vector set to $[0, 1, \ldots, n-1]^T$, where n denotes the order of the linear system. In Table 1, the convergence behavior and the parallel performance of the PPGMRES(m) method, in conjunction with the MPMSC scheme, for the 3D Poisson problem (n = 1,000,000) are presented for various numbers of cores-subdomains. The communication time is relative to the required iterations for convergence to the prescribed tolerance and the number of cores. It can be easily seen that as the number of cores and subdomains increases, the number of required iterations for convergence to the prescribed tolerance decreases for the 3D Poisson problem. In Table 2, the convergence behavior and the parallel performance of the PPGMRES(m) method, in conjunction with the Block Jacobi preconditioner, for the

Table 1: Convergence behavior and parallel performance of the PPGMRES(m) method, in conjunction with the MPMSC scheme, for the 3D Poisson problem (n = 1,000,000).

Poisson 3D	Cores/subdomains				
	64	128	256	512	1024
MPMSC					
Pre-Time	367.38	70.04	21.45	6.31	3.64
Comm-Time	23.54	13.63	10.92	9.81	8.39
GMRES-Time	122.45	73.77	56.45	50.85	41.51
Total-Time	489.84	143.81	77.90	57.16	45.14
Iterations	4 (8)	4 (13)	4 (12)	4 (7)	3 (15)
Aggregates	13	20	40	84	162
Preconditioned Relative Residual	$9.24E-09$	$9.73E-09$	$8.90E-09$	$9.99E-09$	$8.91E-09$
Relative Residual	$1.38E-08$	$1.48E-08$	$1.28E-08$	$1.54E-08$	$1.47E-08$

Table 2: Convergence behavior and parallel performance of the PPGMRES(m) method, in conjunction with the Block Jacobi preconditioner, for the 3D Poisson problem (n = 1,000,000).

Poisson 3D	Cores/subdomains				
	64	128	256	512	1024
Block Jacobi					
Pre-Time	368.53	69.43	21.42	6.26	3.58
Comm-Time	24.14	16.63	12.21	11.72	12.62
GMRES-Time	133.28	90.17	66.13	64.25	61.81
Total-Time	501.81	159.60	87.55	70.51	65.39
Iterations	4 (18)	5 (15)	5 (9)	5 (15)	5 (12)
Preconditioned Relative Residual	$9.52E-09$	$9.33E-09$	$9.81E-09$	$9.85E-09$	$9.90E-09$
Relative Residual	$1.21E-08$	$1.57E-08$	$1.60E-08$	$1.41E-08$	$1.42E-08$

3D Poisson problem (n = 1,000,000) with various numbers of cores-subdomains are given. It should be stated that the MPMSC scheme presents better convergence behavior than the Block Jacobi preconditioning scheme and therefore requires less overall communication time, especially for many cores where the convergence behavior of the Block Jacobi method is downgraded. Therefore, it should be noted that the MPMSC scheme has better scalability than the Block Jacobi preconditioner. In Figure 3, the speedup of the PPGMRES(m) method, in conjunction with the MPMSC scheme, for the 3D Poisson problem (n = 1,000,000) is depicted. In Figure 4, the weak scalability diagram of time versus cores for MPMSC scheme is presented. The order of the linear system increases as the number of cores increases since the local linear system should be of constant order. This results in augmenting, by 10,000 more unknowns, the linear system for every added core. The Total-Time should ideally remain constant. It should be noted that the Pre-Time is almost constant, as the initialization phase of the preconditioner is an inherently parallel process. From Figures 3 and 4 it is evident that the proposed method is scalable up to large number of cores, for the efficient solution of large linear systems. It should be mentioned that partitioning the domain Ω into more subdomains improves the convergence behavior of the method, since the grid corresponding to the aggregated (coarse) components becomes finer, which improves the accuracy of the local approximations to the solution. The improved convergence behavior reduces the communications and hence improves the performance of the proposed scheme. Due to the reduced communications, while the local computational work remains almost constant, it results in superlinear speedups, especially in the case of moderate number of subdomains. For small number of cores the Pre-Time is dominant compared to the GMRES-Time and as the preprocessing phase is an inherently parallel process, both methods have satisfactory speedup. When the number of cores is 1024, the Pre-Time is a small percentage of the Total-Time; therefore the speedup reduces, since it relies on the parallelization of the PPGMRES(m). It should be noted that the GMRES-Time increases proportionally to the problem size, as global communications, related to the synchronization of distributed vectors, and sequential redundant operations of PPGMRES(m) depend on the size of global vectors, compare [28], as it can be seen in Figure 4. However, the inherently parallel preprocessing phase scales, with satisfactory speedup, up to large number cores-subdomains. Thus, the Total-Time required by the proposed scheme is improved.

Furthermore, the convergence behavior of MPMSC scheme for 3D Poisson problem with 10,000 unknowns per subdomain is shown in Figure 5. The total number of iterations is computed as ((outer $*$ m) + inner). It should be stated that the outer iterations of MPMSC scheme are almost constant for large number of subdomains. Moreover, the

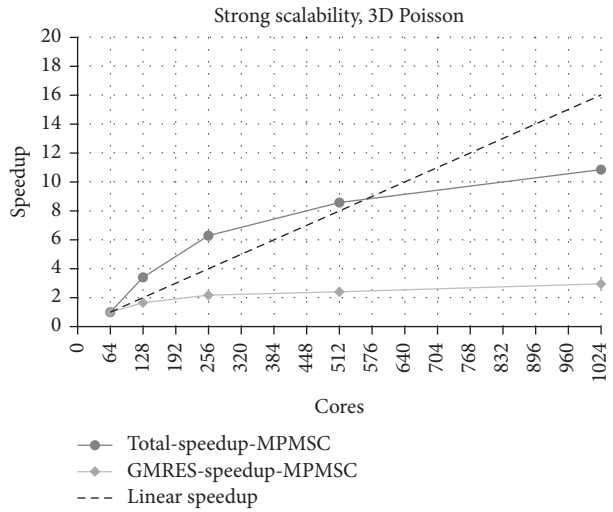

Figure 3: The speedup of the PPGMRES(m) method, in conjunction with the MPMSC scheme, for the 3D Poisson problem (n = 1,000,000).

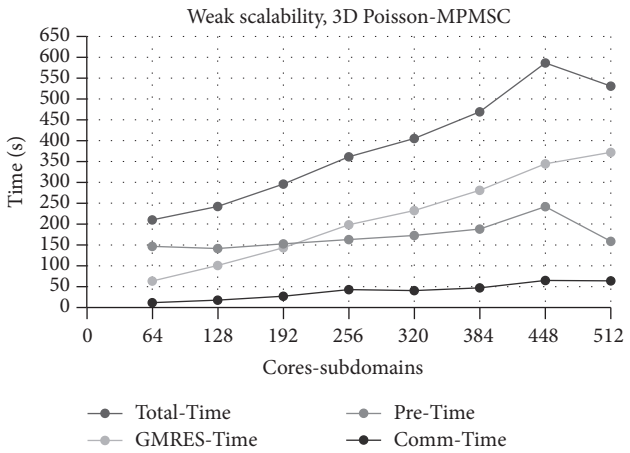

Figure 4: The weak scalability (time versus cores) of the MPMSC scheme for 3D Poisson problem.

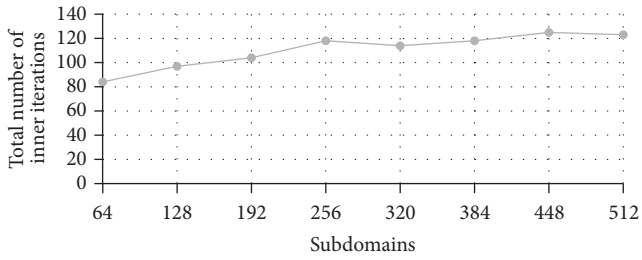

Figure 5: The convergence behavior of MPMSC scheme for 3D Poisson problem with various numbers of subdomains.

convergence behavior of the MPMSC scheme is related to the quality of the graph partitioning, provided by Metis, compare [23]. In Figure 6, the number of aggregates (n_{aggs}) of the MPMSC scheme against the number of subdomains for the 3D Poisson problem with 10,000 unknowns per subdomain is depicted.

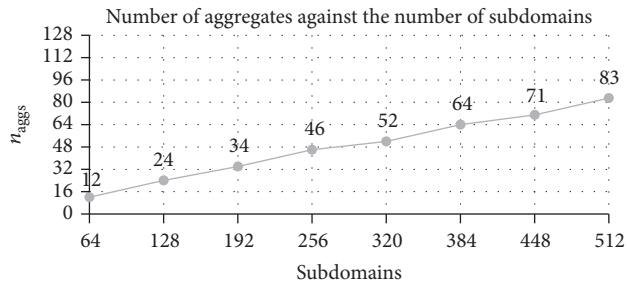

Figure 6: The number of aggregated (coarse) components (n_{aggs}) of the MPMSC scheme for 3D Poisson problem with various numbers of subdomains.

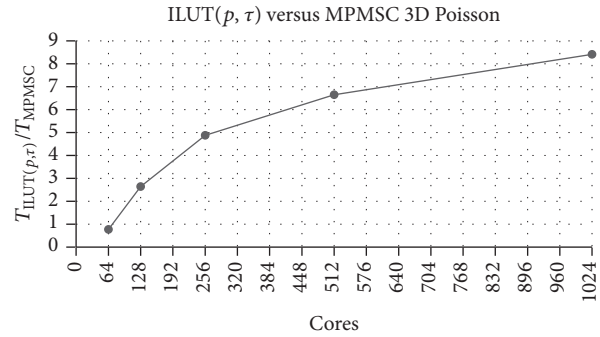

Figure 7: The speedup of PPGMRES(m) in conjunction with MPMSC scheme with respect to the preconditioned GMRES(m) in conjunction with the ILUT(p, τ) preconditioner, with $p = 50$ and $\tau = 10^{-3}$, and various numbers of cores for the 3D Poisson problem (n = 1,000,000).

In Table 3, the performance and convergence behavior of MPMSC scheme with distance two neighborhood for the 3D Poisson problem (n = 1,000,000) with various numbers of cores-subdomains are presented. It should be mentioned that using distance two neighborhood in the subspace compression technique results in less aggregates compared to distance one neighborhood, which decreases the memory requirements. However, since the required number of iterations for convergence increases, the performance of the iterative scheme is downgraded.

Furthermore, the parallel performance of the proposed method is compared to sequential preconditioned GMRES(m) in conjunction with dual threshold incomplete LU factorization (ILUT(p, τ)) preconditioner, compare [1]. The speedup of PPGMRES(m) in conjunction with MPMSC scheme with respect to the preconditioned GMRES(m) in conjunction with the ILUT(p, τ) preconditioner, with $p = 50$ and $\tau = 10^{-3}$, and various numbers of cores, for the 3D Poisson problem (n = 1,000,000), is presented in Figure 7. The preconditioned GMRES(m) in conjunction with the ILUT(p, τ) preconditioner required 2 (14) iterations for convergence to the prescribed tolerance. The PPGMRES(m) in conjunction with the MPMSC scheme performed better, in terms of total execution time, compared to the preconditioned GMRES(m) in conjunction with the ILUT(p, τ)

Table 3: Convergence behavior and parallel performance of the PPGMRES(m) method, in conjunction with the MPMSC scheme with distance two neighborhood for the 3D Poisson problem (n = 1,000,000).

Poisson 3D	Cores/subdomains				
	64	128	256	512	1024
MPMSC $d = 2$					
Pre-Time	368.53	69.43	21.42	6.26	3.58
Comm-Time	24.14	16.63	12.21	11.72	12.62
GMRES-Time	133.28	90.17	66.13	64.25	61.81
Total-Time	501.81	159.60	87.55	70.51	65.39
Iterations	4 (18)	5 (15)	5 (9)	5 (15)	5 (12)
Aggregates	6	9	19	32	51
Preconditioned Relative Residual	$9.18E-09$	$8.62E-09$	$9.57E-09$	$8.80E-09$	$8.56E-09$
Relative Residual	$1.57E-08$	$1.18E-08$	$1.59E-08$	$1.38E-08$	$1.46E-08$

Table 4: Problems from the University of Florida sparse matrix collection.

Problem	Application	Size(A)	nnz(A)
atmosmodd	Atmospheric modeling	1270432	8814880
atmosmodl		1489752	10319760
tmt_sym	Electromagnetics problem	726713	5080961
apache2	Structural problem	715176	4817870
Cage14	DNA electrophoresis	1505785	27130349
ecology2	Ecology problem	999999	4995991
thermal2	Thermal problem	1228045	8580313

preconditioner, especially for large numbers of cores-subdomains.

3.2. UFL Matrices. In order to assess the applicability, scalability, and performance of MPMSC scheme for solving general large sparse linear systems, seven matrices from the University of Florida sparse matrix collection, compare [37], have been considered. Information concerning the aforementioned matrices is given in Table 4. The right-hand-side vectors for atmosmodd, atmosmodl, and thermal2 problems were given by the University of Florida sparse matrix collection, compare [37], whereas, for the rest of the problems, they were computed as the product of coefficient matrix A and the solution vector set to $[0, 1, \ldots, n-1]^T$.

The convergence behavior and the parallel performance of the PPGMRES(m) method, in conjunction with the MPMSC scheme for the atmosmodd, atmosmodl, tmt_sym, apache2, and cage14 problems, are presented in Table 5. In Tables 6 and 7, the convergence behavior and the parallel performance of the PPGMRES(m) method, in conjunction with the MPMSC scheme for the ecology2 and the thermal2 problems, are given. In Figures 8 and 9, the speedup of the PPGMRES(m) method, in conjunction with the MPMSC scheme for the ecology2 and the thermal2 problem, is shown.

For the ecology2 problem, the local coefficient matrices have reduced number of nonzero elements and hence Pre-Time is a small fraction of the Total-Time, not affecting the Total-Speedup curve, since the *LU* factorization requires reduced number of floating point operations. In contrast, for the thermal2 problem compared to ecology2 problem, the local coefficient matrices have more nonzero elements resulting in increased Pre-Time, since the *LU* factorization requires increased number of floating point operations, especially for small number of cores/subdomains. As the number of cores/subdomains increases, the Pre-Time decreases rapidly, because there are less nonzero elements in the local coefficient matrices resulting in improved performance of the *LU* factorization. Therefore, superlinear speedup is achieved for the thermal2 problem and not for the ecology2 problem.

The speedup of PPGMRES(m) in conjunction with MPMSC scheme with respect to the preconditioned GMRES(m) in conjunction with the ILUT(p, τ) preconditioner, with $p = 100$ and $\tau = 10^{-6}$, and various numbers of cores, for the thermal2 problem, is presented in Figure 10. The preconditioned GMRES(m) in conjunction with the ILUT(p, τ) preconditioner required 11 (2) iterations for convergence to the prescribed tolerance. The PPGMRES(m) based on MPMSC scheme performed better, in terms of total execution time, compared to the preconditioned GMRES(m) in conjunction with the ILUT(p, τ) preconditioner, especially for large numbers of cores-subdomains.

It should be noted that the preconditioned GMRES(m) in conjunction with the ILUT(p, τ) preconditioner for the ecology2 problem exceeds the available memory resources and does not converge to the prescribed tolerance for any compatible choice of p or τ not exceeding available memory resources.

Table 5: Convergence behavior and parallel performance of the PPGMRES(m) method, in conjunction with the MPMSC scheme for various problems.

MPMSC	Cores/subdomains				
	64	128	256	512	1024
atmosmodd					
Iterations	5 (12)	6 (12)	6 (10)	6 (17)	7 (10)
Total-Time	759.43	255.30	137.97	108.40	109.37
atmosmodl					
Iterations	1 (10)	1 (16)	1 (15)	1 (15)	1 (16)
Total-Time	439.56	271.09	77.05	43.85	36.38
tmt_sym					
Iterations	22 (17)	19 (7)	13 (18)	12 (16)	11 (18)
Total-Time	272.72	175.99	117.74	105.86	96.67
apache2					
Iterations	8 (19)	8 (15)	7 (20)	7 (16)	5 (20)
Total-Time	189.46	105.15	74.75	66.76	50.49
cage14					
Iterations	—	—	0 (10)	0 (10)	0 (11)
Total-Time	—	—	1,296.23	196.70	62.69

"—": memory limits were reached in the initialization phase.

Table 6: Convergence behavior and parallel performance of the PPGMRES(m) method, in conjunction with the MPMSC scheme for ecology2 problem.

ecology2	Cores/subdomains				
	64	128	256	512	1024
MPMSC					
Pre-Time	20.09	6.60	3.50	2.69	2.51
Comm-Time	84.46	80.90	47.66	32.48	32.33
GMRES-Time	547.96	434.61	260.92	164.42	153.88
Total-Time	568.06	441.21	264.42	167.11	156.40
Iterations	34 (18)	35 (4)	23 (11)	14 (14)	14 (7)
Aggregates	19	39	62	139	263
Preconditioned Relative Residual	$9.70E-09$	$9.26E-09$	$9.96E-09$	$9.64E-09$	$9.91E-09$
Relative Residual	$9.10E-08$	$6.99E-08$	$8.67E-08$	$7.19E-08$	$1.32E-07$

Table 7: Convergence behavior and parallel performance of the PPGMRES(m) method, in conjunction with the MPMSC scheme for thermal2 problem.

thermal2	Cores/subdomains				
	64	128	256	512	1024
MPMSC					
Pre-Time	1491.13	198.63	51.66	10.12	5.51
Comm-Time	176.09	42.41	45.64	44.51	26.97
GMRES-Time	738.37	350.66	297.53	251.73	149.12
Total-Time	2229.50	549.29	349.19	261.84	154.64
Iterations	17 (12)	16 (18)	18 (6)	17 (16)	10 (16)
Aggregates	21	38	70	135	268
Preconditioned Relative Residual	$9.98E-09$	$9.61E-09$	$1.00E-08$	$9.97E-09$	$9.87E-09$
Relative Residual	$5.36E-09$	$9.94E-09$	$5.17E-09$	$5.46E-09$	$7.80E-09$

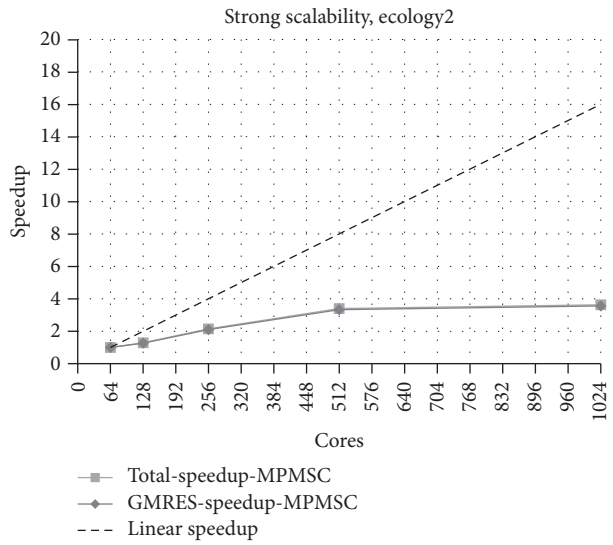

Figure 8: The speedup of the PPGMRES(m) method, in conjunction with the MPMSC scheme, for the ecology2 problem.

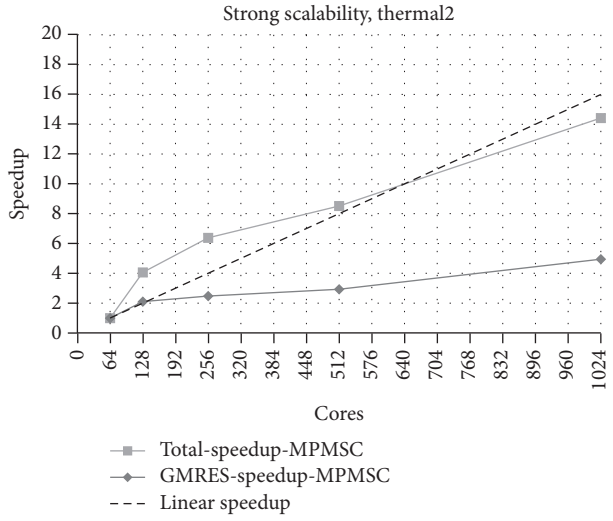

Figure 9: The speedup of the PPGMRES(m) method, in conjunction with the MPMSC scheme, for the thermal2 problem.

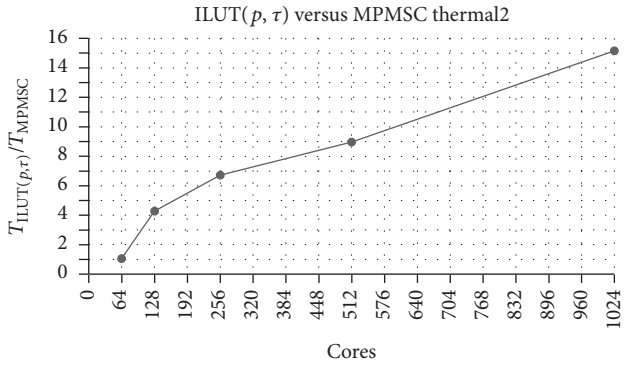

Figure 10: The speedup of PPGMRES(m) in conjunction with MPMSC scheme with respect to the preconditioned GMRES(m) in conjunction with the ILUT(p, τ) preconditioner, with $p = 100$ and $\tau = 10^{-6}$, and various numbers of cores, for the thermal2 problem.

4. Conclusion

The proposed scheme, namely, multiprojection method with subspace compression, has been shown to be effective in solving various problems in distributed memory parallel systems. Further, the PPGMRES(m) in conjunction with the MPMSC scheme scales up to a large number of cores exhibiting superlinear speedups, especially for problems that require large computational work in the initialization phase of the preconditioning scheme. The proposed scheme has improved convergence behavior as the number of subdomains increases compared to the extant domain decomposition based preconditioners. Future work will be concentrated towards using asynchronous and neighboring communications between heterogeneous compute nodes, in order to assess the performance of the method in modern cloud environments. Moreover, the choice of the most suitable parallel graph partitioning algorithm, according to the performance and the partitioning quality, will be studied.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to express their thanks to Professor Dana Petcu, Department of Computer Science, West University of Timisoara (UVT), for the provision of computational facilities through the UVT HPC Center.

References

[1] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 2nd edition, 2003.

[2] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.

[3] J. H. Bramble, J. E. Pasciak, and A. H. Schatz, "The construction of preconditioners for elliptic problems by substructuring *I*," *Mathematics of Computation*, vol. 47, no. 175, pp. 103–134, 1986.

[4] X.-C. Cai and M. Sarkis, "A restricted additive Schwarz preconditioner for general sparse linear systems," *SIAM Journal on Scientific Computing*, vol. 21, no. 2, pp. 792–797, 1999.

[5] L. M. Carvalho, L. Giraud, and G. Meurant, "Local preconditioners for two-level non-overlapping domain decomposition methods," *Numerical Linear Algebra with Applications*, vol. 8, no. 4, pp. 207–227, 2001.

[6] T. F. Chan and T. P. Mathew, "Domain decomposition algorithms," *Acta Numerica*, vol. 3, pp. 61–143, 1994.

[7] M. Dryja, B. F. Smith, and O. B. Widlund, "Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions," *SIAM Journal on Numerical Analysis*, vol. 31, no. 6, pp. 1662–1694, 1994.

[8] B. Smith, P. Björstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.

[9] L. Giraud, F. Guevara Vasquez, and R. S. Tuminaro, "Grid transfer operators for highly variable coefficient problems in

two-level non-overlapping domain decomposition methods," *Numerical Linear Algebra with Applications*, vol. 10, no. 5-6, pp. 467–484, 2003.

[10] M. Manguoglu, A. H. Sameh, and O. Schenk, "PSPIKE: a parallel hybrid sparse linear system solver," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5704, pp. 797–808, 2009.

[11] A. H. Sameh and V. Sarin, "Hybrid parallel linear system solvers," *International Journal of Computational Fluid Dynamics*, vol. 12, no. 3-4, pp. 213–223, 1999.

[12] C. Farhat and F. Roux, "A method of finite element tearing and interconnecting and its parallel solution algorithm," *International Journal for Numerical Methods in Engineering*, vol. 32, no. 6, pp. 1205–1227, 1991.

[13] C. Farhat, M. Lesoinne, and K. Pierson, "A scalable dual-primal domain decomposition method," *Numerical Linear Algebra with Applications*, vol. 7, no. 7-8, pp. 687–714, 2000.

[14] A. Klawonn and O. B. Widlund, "A domain decomposition method with Lagrange multipliers and inexact solvers for linear elasticity," *SIAM Journal on Scientific Computing*, vol. 22, no. 4, pp. 1199–1219, 2000.

[15] J. Mandel, "Balancing domain decomposition," *Communications in Numerical Methods in Engineering with Biomedical Applications*, vol. 9, no. 3, pp. 233–241, 1993.

[16] C. R. Dohrmann, "A preconditioner for substructuring based on constrained energy minimization," *SIAM Journal on Scientific Computing*, vol. 25, no. 1, pp. 246–258, 2003.

[17] E. Agullo, L. Giraud, A. Guermouche, A. Haidar, and J. Roman, "Parallel algebraic domain decomposition solver for the solution of augmented systems," *Advances in Engineering Software*, vol. 60-61, pp. 23–30, 2013.

[18] Y. Zhu and A. H. Sameh, "PSPIKE+: a family of parallel hybrid sparse linear system solvers," *Journal of Computational and Applied Mathematics*, vol. 311, pp. 682–703, 2017.

[19] R. Bank, R. Falgout, T. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, "Algebraic multigrid domain and range decomposition (AMG-DD/AMG-RD)," *SIAM Journal on Scientific Computing*, vol. 37, no. 5, pp. S113–S136, 2015.

[20] R. E. Bank and R. K. Smith, "An algebraic multilevel multigraph algorithm," *SIAM Journal on Scientific Computing*, vol. 23, no. 5, pp. 1572–1592, 2002.

[21] W. F. Mitchell, "A parallel multigrid method using the full domain partition," *Electronic Transactions on Numerical Analysis*, vol. 6, pp. 224–233, 1997.

[22] Y. Xi, R. Li, and Y. Saad, "An algebraic multilevel preconditioner with low-rank corrections for sparse symmetric matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 1, pp. 235–259, 2016.

[23] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[24] F. Pellegrini and J. Roman, "Scotch: a software package for static mapping by dual recursive bipartitioning of process and architecture graphs," in *High-Performance Computing and Networking*, vol. 1067 of *Lecture Notes in Computer Science*, pp. 493–498, Springer, Berlin, Germany, 1996.

[25] H. Kim, J. Xu, and L. Zikatanov, "A multigrid method based on graph matching for convection-diffusion equations," *Numerical Linear Algebra with Applications*, vol. 10, no. 1-2, pp. 181–195, 2003.

[26] Y. Notay, "Aggregation-based algebraic multilevel preconditioning," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 4, pp. 998–1018, 2006.

[27] Y. Notay, "An aggregation-based algebraic multigrid method," *Electronic Transactions on Numerical Analysis*, vol. 37, pp. 123–146, 2010.

[28] C. C. Douglas, G. Haase, and U. Langer, *A Tutorial on Elliptic PDE Solvers and Their Parallelization*, vol. 16 of *Software, Environments, and Tools*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa, USA, 2003.

[29] Y. Saad and M. H. Schultz, "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 1986.

[30] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, vol. 95 of *Applied Mathematical Sciences*, Springer-Verlag, New York, NY, USA, 1994.

[31] A. Frommer and D. B. Szyld, "An algebraic convergence theory for restricted additive Schwarz methods using weighted max norms," *SIAM Journal on Numerical Analysis*, vol. 39, no. 2, pp. 463–479, 2001.

[32] T. A. Davis, *Direct Methods for Sparse Linear Systems*, SIAM, Philadelphia, Pa, USA, 2006.

[33] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, Md, USA, 4th edition, 2013.

[34] B. Chapman, G. Jost, and R. van der Pas, *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*, MIT Press, Cambridge, Mass, USA.

[35] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, "Basic linear algebra subprograms for fortran usage," *ACM Transactions on Mathematical Software (TOMS)*, vol. 5, no. 3, pp. 308–323, 1979.

[36] G. Karypis and V. Kumar, "A parallel algorithm for multilevel graph partitioning and sparse matrix ordering," *Journal of Parallel and Distributed Computing*, vol. 48, no. 1, pp. 71–95, 1998.

[37] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *Association for Computing Machinery. Transactions on Mathematical Software*, vol. 38, no. 1, 25 pages, 2011.