Fast Direct LDL' Solver for Method of Moments Electric Field Integral Equation Solution

Yoginder Kumar Negi[1], N. Balakrishnan[1], and Sadasiva M. Rao[2]

[1]Supercomputer Education and Research Center
Indian Institute of Science, Bangalore 560012, India
yknegi@gmail.com, balki@iisc.ac.in

[2]Naval Research Laboratory
Washington DC 20375, USA
sadasiva.rao@nrl.navy.mil

Abstract – This paper proposes a new fast direct solver using the block diagonalization method. In our proposed method, the symmetric half single-level compressed block matrix is factorized using the diagonalization method into block diagonal and upper triangle block $\boldsymbol{LDL'}$ format where, due to symmetric property, $\boldsymbol{L}$ is a transpose of $\boldsymbol{L'}$. The far-field blocks in the upper triangle row block are merged and compressed using Adaptive Cross Approximation (ACA) and $\boldsymbol{QR}$ factorization. The solution consists of solving the diagonal block matrix and matrix-vector multiplication of the compressed row blocks of the upper triangle matrices. Our results show that the factorization cost and memory scales to $O(N^{1.5})$ and the solution process scales to $O(N)$. The method generates an efficient solution process for solving large-scale electromagnetics problems.

Index Terms – Adaptive Cross Approximation (ACA), Electric Field Integral Equation (EFIE), electromagnetics scattering, fast direct solver, matrix compression, Method of Moments (MoM).

I. INTRODUCTION

Method of Moments (MoM) is a well-known integral equation-based Computational Electromagnetics (CEM) [1] method for solving complex electromagnetic radiation and scattering problems numerically in the frequency domain. Compared to differential equation-based methods like Finite Element Method (FEM) [2] and Finite Difference Time Domain (FDTD) [3] methods, MoM is free from grid dispersion error and leads to a smaller matrix size than FEM. The MoM application for solving large-scale electromagnetics problems is limited by dense matrix computation property with the matrix computation and storage cost of $O(N^2)$ for N number of unknowns. Solving the MoM dense matrix with a direct solver leads to $O(N^3)$ computation cost and with iterative solver leads to N_{itr} $O(N^2)$ cost for N_{itr} iteration count with $O(N^2)$ matrix-vector product cost. The high matrix computation, storage, and solution cost is mitigated by various matrix compression-based fast solver algorithms proposed by various researchers. The matrix compression for fast solvers may be of two categories: analytical-based compression and algebraic matrix compression. Examples of analytic-based compression methods are Multilevel Fast Multipole Algorithm (MLFMA) [4], Adaptive Integral Method (AIM) [5], and pre-corrected FFT [6]. Similarly, we have algebraic matrix compressed methods like Adaptive Cross Approximation (ACA) [7, 8], IE-QR [9], and H-Matrix [10–12]. Analytical fast solvers are kernel-dependent, whereas algebraic fast solvers are kernel-independent and easy to implement. All the fast solvers work on reducing matrix filling, solution, and storage time. Full wave fast solver in CEM reduces matrix filling and storage time to $O(NlogN)$. However, the matrix solution time depends on the method of solution and whether the algorithm adopts an iterative approach or a direct approach. For the iterative solution, the solution time scales as N_{itr} $O(NlogN)$ for $O(NlogN)$ compressed matrix-vector product cost. The solution of these methods relies on the compressed matrix's iterative solution. The iterative solution process is a convergence-dependent method for each matrix-vector product iteration. Furthermore, the convergence for each iteration depends on the condition number of the matrix computed. It is well-known that the Electric Field Integral Equation (EFIE) [13] gives an ill-conditioned MoM matrix. This ill-conditioning due to the closed geometry structure can be overcome by using a Magnetic Field Integral Equation (MFIE) [14] and a Combined Field Integral Equation (CFIE) [15]. Also, the ill-conditioning may be due to structural geometry or mesh quality. The ill-conditioning leads to a high

Submitted On: December 16, 2024
Accepted On: March 26, 2025
https://doi.org/10.13052/2025.ACES.J.400401

iteration count when solved with an iterative solver. The high iteration is mitigated by using various matrix preconditioners [16–18]. The preconditioner computation comes with the extra precondition computation cost and precondition solution cost during the iterative solution process.

Solving multiple right-hand side (RHS) electromagnetics problems like in monostatic radar cross section (RCS) computation or multiport microwave network system analysis with a fast iterative solver may lead to high solution time. The overall solution time will scale to N_{rhs} N_{itr} $O(NlogN)$ for N_{rhs} RHS. Also, for a multi-RHS or single-RHS system, the number of unknowns increases with the increase in simulation frequency or geometry size. With the increase in the number of unknowns, solving the linear system of equations with an iterative, fast solver will lead to a further increase in the number of iterations (N_{itr}) for the solution process.

The lacuna of the iterative solver for solving linear systems of equations is overcome by using direct matrix solution methods. Direct solvers are the most reliable method for solving any linear system of equations, with a guaranteed solution for the system. However, the high factorization and solution costs hinder the application of direct solvers for significant electromagnetic problems. In the past decade, there has been an inclination among the fast solver research community to develop direct solution-based fast solvers. MLFMA-based analytical fast direct solvers proposed in [19, 20] are kernel-dependent. Algebraic fast direct solvers built upon extended matrix [21] and H^2-Matrix [22] does not scale well for significant problems. A power series-based fast direct solver is proposed in [23, 24], where the solution convergence depends on the matrix's condition number. Sherman-Morrison-Woodbury based fast direct solvers [25, 26] have high factorization and solution time. A ***LU***-based fast direct solver is proposed in [27–29], and factorization is applied to a single-level compressed MoM matrix. ***LU*** matrix factorization and solutions are serial and difficult to parallelize.

This work proposes a new fast direct solver based on the diagonalization applied on a symmetric half single-level compressed block MoM matrix. The factorization cost is reduced by applying low-rank block matrix operation, and the linear cost of the matrix solution is achieved by merging the compressed far-field matrix blocks into a single compressed matrix block. The solution process consists of block diagonal matrix solution and matrix-vector product of the row block compress matrices. The paper is organized as follows: section II presents a brief description of EFIE MoM with block matrix compression. In section III, we present the proposed block diagonalized fast direct solver. Section III also presents the low-rank matrix operation performed on the compressed matrices to reduce the factorization and solution time. In section IV, the efficiency and accuracy of the proposed fast direct solver is presented. Section V concludes the paper.

II. BRIEF DESCRIPTION OF EFIE-MOM

The MoM matrix is computed for 3D surfaces using EFIE, MFIE, or CFIE formulations. Selection of the integral equation method is essential when solving the matrix with a regular iterative solver. MFIE is only applicable for closed-body geometries. CFIE is a combination of EFIE and MFIE, which further increases matrix computation time. For the sake of clarity, this work uses only EFIE for MoM matrix computation to solve 3D Perfect Electric Conductor (PEC) geometry and can be easily extended to MFIE and CFIE. The governing equation for EFIE is:

$$E_{total} = E_{inc} + E_{scatt} \ , \tag{1}$$

where E_{total} is the total electric field equal to the sum of the incident electric field (E_{inc}) and the scattered electric field (E_{scatt}). Applying the boundary condition, expanding current density ($J(r')$) and charge density ($\rho(r')$) for the electric field vector potential and scalar potential with the RWG [30] basis functions (f_i), and performing Galerkin testing, the elements of the MOM matrix are:

$$\boldsymbol{Z}(i,j) = \frac{j\omega\mu}{4\pi}\iint f_i \cdot f_j \frac{e^{-jk|r-r'|}}{r-r'} dsdt + \frac{1}{j\omega 4\pi\varepsilon}\iint \nabla \\ f_i \frac{e^{-jk|r-r'|}}{r-r'} \nabla \cdot f_j dsdt. \tag{2}$$

In the above equation, the MoM matrix element is computed for the i^{th} test j^{th} source basis. In equation (2), k is the wavenumber, r and r' represent the observer and source points, and μ and ε are the permeability and permittivity of the background material. Integration is performed over the RWG source and testing domains. MoM matrix computation using equation (2) leads to a linear system of equations. The system of equations can be written as a combination of a near-field interaction matrix $\boldsymbol{Z_N}$ and a far-field interaction matrix $\boldsymbol{Z_F}$ given by:

$$[\boldsymbol{Z_N} + \boldsymbol{Z_F}\ ]\boldsymbol{x} = \boldsymbol{b}. \tag{3}$$

Solving equation (3) for $\boldsymbol{x}$ presents computation time and memory limitations, which can be overcome by applying various fast solver methods. These methods work on the compressibility of the far-field matrix.

For the computation of a fast solver, the mesh of the 3D geometry is divided into small blocks using the multilevel binary-tree or oct-tree partition method [12]. The fast solver is the single level when the far-field matrix compression is carried out at the lowest level, and the multilevel is when the matrix compression is done at all levels. Single-level fast solver reduces matrix filling and solution time to $O(N^{1.5})$ whereas multilevel reduces the time complexity to $O(NlogN)$. This work uses a single-level compressed MoM block matrix fast solver to ease block matrix operation. For the single-level fast solver matrix construction, matrix compression is applied at the lowest level of binary-tree-based 3D geometry decomposition, and the interaction is computed for the mesh block satisfying the admissibility condition [12]. At the leaf level, the block interaction that does not satisfy the admissibility condition is considered a near-field interaction. Further, this work employs the re-compressed ACA method [31] to compute a symmetric single-level compressed block matrix. The compressed matrix is made symmetric by averaging the upper diagonal and lower diagonal values and replacing the upper diagonal and lower diagonal values with the average value. Using the symmetric property, we can reuse the upper diagonal value for matrix-vector product and matrix factorization. Exploiting the symmetric property, we compute and save the diagonal and upper diagonal matrix in the compressed far-field and dense near-field blocks [12]. The single-level symmetric half-compressed block matrix is used for factorization and solution process. The formulation for the new fast direct solver with reduced factorization and solution time is discussed in the next section.

III. FAST DIRECT SOLVER

This section discusses the factorization and solution process for developing a fast direct solver applied on a single-level compressed block matrix. The matrix diagonalization process has previously been used to solve the linear system of equations for sparse and dense matrices [32–34]. However, these methods lead to a high cost of factorization and solution. In our previous work [18, 23, 24], the diagonalization of the near-field block matrix is discussed in detail. There, the computation is performed on the dense near-field block matrices. Due to the dense nature of the near-field matrix, the cost of computation and storage is kept low. Extending the diagonalization method to a fast solver-based full matrix is limited by the far-field matrix blocks. In this work, we diagonalize the full MoM compressed matrix. To understand the complete implementation of the factorization process, we will discuss the factorization process for the whole matrix in detail. The far-field matrix operation cost is reduced by performing low-rank block matrix operation. The method is applied to the MoM matrix to factorize it to $\boldsymbol{LDL'}$ format, where $\boldsymbol{D}$ is a diagonal block matrix and $\boldsymbol{L}$ and $\boldsymbol{L}'$ are lower and upper triangle block matrices. The lower triangle block matrix $\boldsymbol{L}$ is the transpose of the upper triangle block matrix, leading to computation and memory savings. In the following subsection, details of the diagonalization process, along with the ways to make it faster for set-up and solution matrix-vector product, are presented.

A. Block matrix factorization

Gaussian Elimination is a well-known method for diagonalizing dense or sparse block matrices. The Gaussian Elimination is performed over the single-level symmetric compressed block matrix. An asymmetric compressed block matrix is computed for the mesh geometry divided into mesh element clusters based on the multilevel binary-tree division. The single-level compressed block matrix near-field and far-field matrix interaction is computed for binary-tree mesh blocks at the lowest level. The symmetric half-block matrix $[\boldsymbol{Z}]$ for factorization is shown below:

$$[\boldsymbol{Z}] = \begin{bmatrix} \boldsymbol{Z}_{11} & \boldsymbol{Z}_{12} & \boldsymbol{Z}_{13} & \boldsymbol{Z}_{14} \\ 0 & \boldsymbol{Z}_{22} & \boldsymbol{Z}_{23} & \boldsymbol{Z}_{24} \\ 0 & 0 & \boldsymbol{Z}_{33} & \boldsymbol{Z}_{34} \\ 0 & 0 & 0 & \boldsymbol{Z}_{44} \end{bmatrix}. \tag{4}$$

The above block matrix consists of near- and far-field matrix blocks. The factorization process consists of diagonalizing the above matrix by multiplying it with the right sparse block matrix. The right scaling matrix nullifies the row blocks of the matrix, leaving a diagonal block matrix. Right scaling matrix $[\boldsymbol{\alpha_1}]$ for first row blocks is given as:

$$[\boldsymbol{\alpha}_1] = \begin{bmatrix} \boldsymbol{I}_{11} & -\boldsymbol{Z}_{11}^{-1}\boldsymbol{Z}_{12} & -\boldsymbol{Z}_{11}^{-1}\boldsymbol{Z}_{13} & -\boldsymbol{Z}_{11}^{-1}\boldsymbol{Z}_{14} \\ 0 & \boldsymbol{I}_{22} & 0 & 0 \\ 0 & 0 & \boldsymbol{I}_{33} & 0 \\ 0 & 0 & 0 & \boldsymbol{I}_{44} \end{bmatrix}, \tag{5}$$

where $\boldsymbol{I}_{11}$, $\boldsymbol{I}_{22}$, $\boldsymbol{I}_{33}$, and $\boldsymbol{I}_{44}$ are the identity block matrices. Equations (4) and (5) are combined to scale the first-row blocks of $[\boldsymbol{Z}]$ to diagonal blocks, and the system of the equation is given as:

$$[\widetilde{\boldsymbol{Z}}^1] = \begin{bmatrix} \boldsymbol{Z}_{11} & \boldsymbol{Z}_{12} & \boldsymbol{Z}_{13} & \boldsymbol{Z}_{14} \\ 0 & \boldsymbol{Z}_{22} & \boldsymbol{Z}_{23} & \boldsymbol{Z}_{24} \\ 0 & 0 & \boldsymbol{Z}_{33} & \boldsymbol{Z}_{34} \\ 0 & 0 & 0 & \boldsymbol{Z}_{44} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{I}_{11} & -\boldsymbol{Z}_{11}^{-1}\boldsymbol{Z}_{12} & -\boldsymbol{Z}_{11}^{-1}\boldsymbol{Z}_{13} & -\boldsymbol{Z}_{11}^{-1}\boldsymbol{Z}_{14} \\ 0 & \boldsymbol{I}_{22} & 0 & 0 \\ 0 & 0 & \boldsymbol{I}_{33} & 0 \\ 0 & 0 & 0 & \boldsymbol{I}_{44} \end{bmatrix}. \tag{6}$$

Equation (6) is represented as:

$$[\widetilde{\boldsymbol{Z}}^1] = [\boldsymbol{Z}]\,[\boldsymbol{\alpha}_1]. \tag{7}$$

Performing the block matrix and scaling matrix multiplication in the above equation, the first-row block diagonalized matrix block equation is written as:

$$\left[\widetilde{\mathbf{Z}}^1\right] = \begin{bmatrix} \mathbf{Z}_{11} & 0 & 0 & 0 \\ 0 & \mathbf{Z}_{22}-\mathbf{Z}_{21}\mathbf{Z}_{11}^{-1}\mathbf{Z}_{12} & \mathbf{Z}_{23}-\mathbf{Z}_{21}\mathbf{Z}_{11}^{-1}\mathbf{Z}_{13} & \mathbf{Z}_{24}-\mathbf{Z}_{21}\mathbf{Z}_{11}^{-1}\mathbf{Z}_{14} \\ 0 & 0 & \mathbf{Z}_{33}-\mathbf{Z}_{31}\mathbf{Z}_{11}^{-1}\mathbf{Z}_{13} & \mathbf{Z}_{34}-\mathbf{Z}_{31}\mathbf{Z}_{11}^{-1}\mathbf{Z}_{14} \\ 0 & 0 & 0 & \mathbf{Z}_{44}-\mathbf{Z}_{41}\mathbf{Z}_{11}^{-1}\mathbf{Z}_{14} \end{bmatrix}. \tag{8}$$

Equation (8) gives the diagonalization of the first-row blocks. In the above equation, using the symmetry property, the matrix blocks $\boldsymbol{Z}_{21}$, $\boldsymbol{Z}_{31}$, and $\boldsymbol{Z}_{41}$ are obtained by computing the transpose of $\boldsymbol{Z}_{12}$, $\boldsymbol{Z}_{13}$, and $\boldsymbol{Z}_{14}$. So $\boldsymbol{Z}_{21} = \boldsymbol{Z'}_{12}$, $\boldsymbol{Z}_{31} = \boldsymbol{Z'}_{13}$, and $\boldsymbol{Z}_{41} = \boldsymbol{Z'}_{14}$. Likewise, each row block is diagonalized by computing the row scaling matrix block and multiplying it with the row diagonalized matrix. The final diagonalization process is given as:

$$[\boldsymbol{Z_D}] = [\boldsymbol{\alpha}_3^{\mathrm{T}}][\boldsymbol{\alpha}_2^{\mathrm{T}}][\boldsymbol{\alpha}_1^{\mathrm{T}}][Z][\boldsymbol{\alpha}_1][\boldsymbol{\alpha}_2]][\boldsymbol{\alpha}_3] \,. \tag{9}$$

Equation (9) written in diagonal form is:

$$[\boldsymbol{Z_D}] = \begin{bmatrix} \boldsymbol{Z}_{11} & 0 & 0 & 0 \\ 0 & \widetilde{\boldsymbol{Z}}_{22} & 0 & 0 \\ 0 & 0 & \widetilde{\boldsymbol{Z}}_{33} & 0 \\ 0 & 0 & 0 & \widetilde{\boldsymbol{Z}}_{44} \end{bmatrix} . \tag{10}$$

In equation (10), $[\boldsymbol{Z_D}]$ is the diagonal block matrix $\boldsymbol{D}$ and $[\boldsymbol{\alpha}_1][\boldsymbol{\alpha}_2]][\boldsymbol{\alpha}_3]$ is the upper triangle block matrix $\boldsymbol{U}$ of the $\boldsymbol{LDU}$ factorization. $\boldsymbol{L}$ is the transpose of $\boldsymbol{U}$ due to the symmetric property of the matrix. The block diagonalized system of the equation is written as:

$$[\boldsymbol{Z_D}]\,[\widetilde{\boldsymbol{x}}] = \left[\widetilde{\boldsymbol{b}}\right] . \tag{11}$$

Here, $[\boldsymbol{b}]$ and $[\boldsymbol{x}]$ is computed by:

$$\left[\widetilde{\boldsymbol{b}}\right] = [\boldsymbol{\alpha}_3^{\mathrm{T}}][\boldsymbol{\alpha}_2^{\mathrm{T}}][\boldsymbol{\alpha}_1^{\mathrm{T}}][\boldsymbol{b}], \tag{12}$$

$$[\boldsymbol{x}]P = [\boldsymbol{\alpha}_1][\boldsymbol{\alpha}_2][\boldsymbol{\alpha}_3][\widetilde{\boldsymbol{x}}]. \tag{13}$$

The final solution process consists of solving equation (11) and performing block matrix-vector products to extract the solution vector in equations (12) and (13). The matrix solution cost reduction will be discussed later.

The major cost of the above-discussed process is the factorization process. The process includes the computation of the block scaling matrix in equation (5) and performing block matrix operations given in equation (8). The generalized matrix operation in equation (8) for scaling k^{th} row block operating on varying m^{th} row and n^{th} column matrix blocks is written as:

$$\widetilde{\boldsymbol{Z}}_{mn} = \boldsymbol{Z}_{mn} - \boldsymbol{Z}_{mk}\boldsymbol{Z}_{kk}^{-1}\boldsymbol{Z}_{kn}. \tag{14}$$

In equation (14), for nullifying one row, there are four major matrix operations to be performed on the remaining matrix blocks. The matrix operations are: matrix inversion of the diagonal block matrix $[\boldsymbol{Z}_{kk}^{-1}]$, matrix solution for block matrix $[\boldsymbol{Z}_{kn}]$, matrix multiplication, and block matrix addition. To perform the above operations in equation (8), matrix inversion needs to be computed once; a matrix block solution is required for the selected row block to be nullified. The major matrix operations to be repeated for each block computation are matrix multiplication and addition. When all the blocks are dense, the operations are straightforward, but this will lead to a high factorization cost, along with a high matrix computation cost. In this work, we reduced the cost of matrix computation and solution as follows.

The matrix consists of dense near-field matrix blocks and low-rank compressed far-field matrix blocks, as given in equation (3). The near-field dense matrix operations are performed using equation (14). However, to keep the computation cost low, we will perform a low-rank block matrix operation [35] on the far-field compressed form matrix. From equation (14), the low-rank matrix operating will be required for block matrix solution for the computation of scaling matrix, block matrix multiplication, and block matrix addition. The low-rank matrix operations are discussed in further subsections.

1. Low-rank matrix solution

The low-rank matrix solution is computed in equation (14) for m^{th} row and n^{th} column block as $[\boldsymbol{Z}_{kk}^{-1}][\boldsymbol{Z}_{kn}]$, where $[\boldsymbol{Z}_{kk}^{-1}]$ is $\boldsymbol{LU}$ factorized dense matrix and $[\boldsymbol{Z}_{kn}]$ is a low-rank compressed matrix of the form $[\boldsymbol{U}]_{kn}[\boldsymbol{V}]_{kn}$, and $[\boldsymbol{U}]_{kn}$ is of size $u \times r$ and $[\boldsymbol{V}]_{kn}$ is of size $r \times v$, for u row size, v column size, and r rank of the block matrix such that $r'\ min(m,n)$. Solving $[\boldsymbol{Z}_{kk}^{-1}]$ for $[\boldsymbol{U}]_{kn}$ and multiplying with $[\boldsymbol{V}]_{kn}$ is equal to solving $[\boldsymbol{Z}_{kk}^{-1}]$ for $[\boldsymbol{Z}_{kn}]$ as shown in Fig. 1. The process saves the solution time by avoiding a full matrix solution.

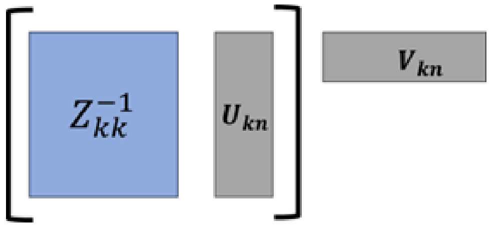

Fig. 1. Low-rank block matrix solution. The matrix operation inside the brackets is performed first to reduce the computation cost.

2. Low-rank matrix multiplication

Low-rank matrix multiplication is computed in equation (14) for m^{th} row and n^{th}column block for $[\boldsymbol{Z}_{mk}]$ and $[\boldsymbol{Z}_{kk}^{-1}\boldsymbol{Z}_{kn}]$. The low-rank multiplication in this operation comes with two scenarios. First, $[\boldsymbol{Z}_{mk}]$ is dense and $[\boldsymbol{Z}_{kk}^{-1}\boldsymbol{Z}_{kn}]$ is in compressed form and, second, both $[\boldsymbol{Z}_{mk}]$ and $[\boldsymbol{Z}_{kk}^{-1}\boldsymbol{Z}_{kn}]$ are in compressed form.

In the first case, let us consider $[\boldsymbol{Z}_{mk}]$ of size $u2 \times u1$ and $[\boldsymbol{Z}_{kk}^{-1}\boldsymbol{Z}_{kn}]$ is in the compressed form of $[\boldsymbol{U}1]_{u1\times r1}[\boldsymbol{V}1]_{r1\times v1}$ where $u1$ is row size, $v1$ is column size, and $r1$ is the rank of the block matrix. Compressed fast multiplication is carried out by multiplying $[\boldsymbol{Z}_{mk}]$ with $[\boldsymbol{U}_1]$ and replacing it with $[\boldsymbol{U}_1]$. Multiplication cost is reduced to $u2 \times u1 \times r1$. Similarly, compressed matrix multiplication is performed when $[\boldsymbol{Z}_{mk}]$ is in compressed form and $[\boldsymbol{Z}_{kk}^{-1}\boldsymbol{Z}_{kn}]$ is in dense form.

In the second case, when $[\boldsymbol{Z}_{mk}]$ and $[\boldsymbol{Z}_{kk}^{-1}\boldsymbol{Z}_{kn}]$ both are in compressed form, fast matrix multiplication is carried out by multiplying compressed parts. Let $[\boldsymbol{Z}_{mk}]$ be the compressed form of $[\boldsymbol{U}_1]_{u1\times r1}[\boldsymbol{V}_1]_{r1\times v1}$ where $u1$ is row size, $v1$ is the column size, and $r1$ is the rank of the matrix. Let $[\boldsymbol{Z}_{mk}]$ be the compressed form of $[\boldsymbol{U}_2]_{u2\times r2}[\boldsymbol{V}_2]_{r2\times v2}$ where $u2$ is row size, $v2$ is column size, and $r2$ is the rank of the matrix block. Fast matrix multiplication is carried out by multiplying $[\boldsymbol{V}_1]_{r1\times v1}$ with $[\boldsymbol{U}_2]_{u2\times r2}$ as shown in Fig. 2, leading to a block matrix of the size $r1 \times r2$. The low-rank matrix is multiplied by row block $[\boldsymbol{U}_1]_{u1\times r1}$ or column block $[\boldsymbol{V}_2]_{r2\times v2}$ and replacing the row or column block.

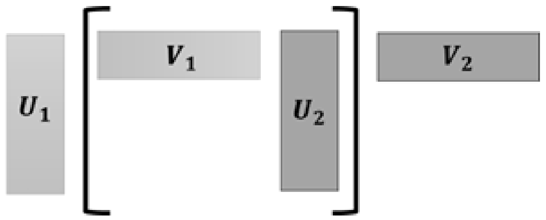

Fig. 2. Low-rank block matrix multiplication operation. Multiplication inside the brackets is performed first for cost saving.

3. Low-rank matrix addition

Low-rank block matrix addition computation for m^{th} row is required for adding matrix blocks $[\boldsymbol{Z}_{mn}]$ and $-[\boldsymbol{Z}_{mk}\boldsymbol{Z}_{kk}^{-1}\boldsymbol{Z}_{kn}]$ in equation (14). Converting the compressed matrices in dense form and adding them will lead to high cost, and adding only the respective compressed row and column blocks leads to an incorrect reluctant matrix. Low-rank matrix operation is performed by merging the respective column row and column blocks and compressing them. Let $[\boldsymbol{Z}_{mn}]$ be of the form $[\boldsymbol{U}_1]_{u1\times r1}[\boldsymbol{V}_1]_{r1\times v1}$ and $-[\boldsymbol{Z}_{mk}\boldsymbol{Z}_{kk}^{-1}\boldsymbol{Z}_{kn}]$ of the form $[\boldsymbol{U}_2]_{u1\times r2}[\boldsymbol{V}_2]_{r2\times v1}$. Then $[\boldsymbol{U}_1]$ and $[\boldsymbol{U}_2]$ is merged by column, and $[\boldsymbol{V}_1]$ and $[\boldsymbol{V}_2]$ is merged by row as shown in Fig. 3. The merged matrices are low-rank matrices and are further compressed and converted into row and column blocks by following the matrix multiplication operation discussed in the previous step.

The low-rank matrix operation discussed above reduces the factorization cost for the full single-level compressed matrix. Factorization is carried out with a row block-wise operation by replacing the existing compressed and dense matrices. This process leads to memory savings with no extra storage requirements for matrix storage. The single-level compressed and dense matrix is replaced by a factorized compressed and dense matrix. The solution time for the factorization can be further reduced by merging the compressed factorized far-field matrix blocks, which are discussed later.

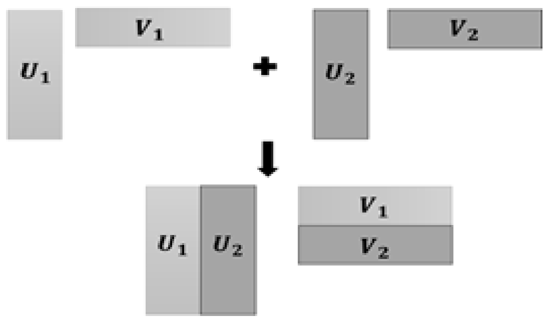

Fig. 3. Low-rank block matrix row and column merging are used to perform low-rank block matrix addition operations.

B. Fast matrix solution

The solution process includes solving the diagonal block matrix in equation (11) and the matrix-vector product in equations (12) and (13). The solution cost of the diagonal block is $O(N)$, as shown in [18], where it is used as a preconditioner. The scaling row block matrices $[\boldsymbol{\alpha}_1]$, $[\boldsymbol{\alpha}_1]$, and $[\boldsymbol{\alpha}_3]$ consist of dense near-field and compressed far-field blocks and for the compressed far-field and dense near-field matrix-vector product cost scales to $O(N^{1.5})$. The matrix-vector product cost is further reduced by merging the far-field compressed blocks as done in H^2-Matrix computation. In the scaling matrix block equation (6), let us suppose all matrix blocks $[-\boldsymbol{Z}_{11}^{-1}\boldsymbol{Z}_{12}]$, $[-\boldsymbol{Z}_{11}^{-1}\boldsymbol{Z}_{13}]$, and $[-\boldsymbol{Z}_{11}^{-1}\boldsymbol{Z}_{14}]$ are in compressed form and are represented as $[\boldsymbol{U}_1]_{u1\times r1}[\boldsymbol{V}_1]_{r1\times v1}$, $[\boldsymbol{U}_2]_{u2\times r2}[\boldsymbol{V}_2]_{r2\times v2}$, and $[\boldsymbol{U}_3]_{u3\times r3}[\boldsymbol{V}_3]_{r3\times v3}$ of the same row basis and size such that $u1 = u2 = u3$. For merging, we follow the following steps:

1. Compute $\boldsymbol{QR}$ decomposition of the first column block matrix. $[\boldsymbol{U}_1]_{u1\times r1}$ into $[\boldsymbol{Q}_1]_{u1\times r1}[\boldsymbol{R}_1]_{r1\times r1}$ where $\boldsymbol{Q}_1$ is an orthonormal matrix and $\boldsymbol{R}_1$ is an upper triangle block matrix.
2. The first compressed row block matrix $[\boldsymbol{V}_1]_{r1\times v1}$ is replaced with the new matrix product $[\boldsymbol{R}_1] \times [\boldsymbol{V}_1]$ represented as $\left[\widetilde{\boldsymbol{V}}_1\right]_{r1\times v1}$.
3. The second matrix block $[\boldsymbol{V}_2]_{r2\times v2}$ is scaled to $[\boldsymbol{Q}_1]$ using transfer matrix computed as $[\boldsymbol{Q}_1]^{-1}\ [\boldsymbol{U}_2]$. As $[\boldsymbol{Q}_1]$ is an orthonormal matrix, the solution process will be $[\boldsymbol{Q}_1]'\ [\boldsymbol{U}_2]$. The transfer matrix will be of size $r1 \times r2$ and is multiplied with $[\boldsymbol{V}_2]_{r2\times v2}$ represented as $\left[\widetilde{\boldsymbol{V}}_2\right]_{r1\times v2}$.

Following the above steps, the third block compressed row matrix $[\boldsymbol{V}_3]_{r3\times v3}$ is scaled to the first block using a transfer matrix and is represented as $\left[\widetilde{\boldsymbol{V}}_3\right]_{r1\times v3}$.The conversion process is pictorially represented in Fig. 4.

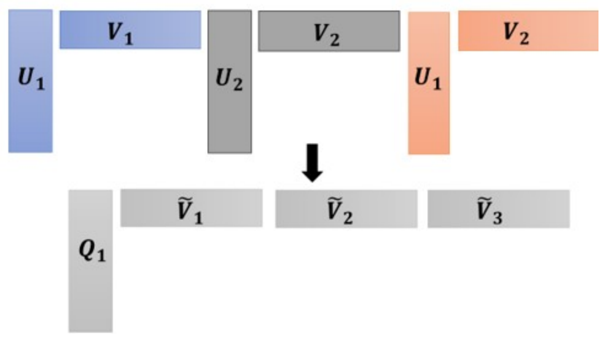

Fig. 4. Conversion of multiple compressed far-field blocks to a single compressed block to reduce matrix-vector product cost.

The new scaling matrix requires storage of the column block matrix. $[Q_1]$ and row block matrices $\left[\widetilde{V}_1\right]$, $\left[\widetilde{V}_2\right]$, and $\left[\widetilde{V}_3\right]$. The row block matrix is merged to form a new matrix $\left[\widetilde{V}_{\text{new}}\right]$. The final compressed matrix will be of the form $[Q_1] \times \left[\widetilde{V}_{new}\right]$. This reduces the overall matrix storage and matrix-vector product cost. Our results show the matrix-vector product cost scaled to $O(N)$.

C. Complexity analysis

Complexity analysis for the near field factorization is discussed in [18]. In this section we will add the far-field operation cost on the proposed method. The proposed algorithm has two steps. First is the low rank compressed single level matrix factorization using low rank matrix operation and second is low rank matrix merging and solution using matrix-vector product. Let the low rank matrix be of size of rank k with row and column size n such that $k << n$.

Step 1. The factorization part consists of low rank matrix solutions. The solution cost for the low rank matrix will be reduced to kn, as the solution is carried out for compressed column matrix. The solution is followed by low rank matrix multiplication operation, where the matrix is merged to form a matrix of size k^2n which is further compressed with complexity of $k^2(n+n)$, retaining the low cost of operation.

Step 2. The merging part includes transpose of the low rank row block and multiplication. The transpose is of $O(1)$ complexity whereas multiplication cost is k^2n. The solution part is of low rank matrix-vector product costing $k^2\,(n+n)$.

Our experimental results for a hollow cylinder of radius 0.25λ with varying lengths and unknowns show that the factorization cost remains to be $O(N^{1.5})$ complexity and total solution time costs maintain $O(N)$ complexity. The complexity is shown in Fig. 5 for an increasing number of unknowns. The cylinder is meshed with tringles for $\lambda/10$ edge length. The binary tree division for mesh is terminated for a block size of 10λ-15λ at the lowest level.

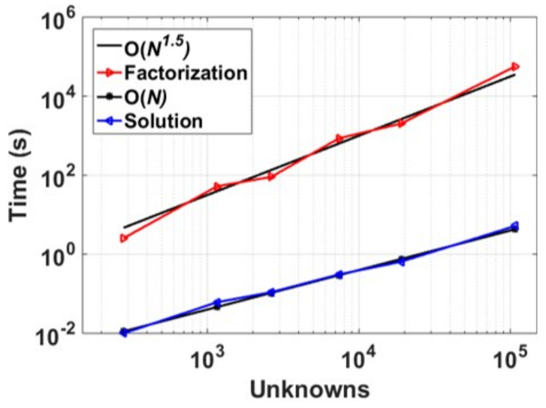

Fig. 5. Block matrix factorization and upper triangle block time using an increasing number of unknowns.

Figure 6 shows memory complexity for factorization process with an increasing number of unknowns. It is observed that the process gives $O(N^{1.5})$ memory complexity.

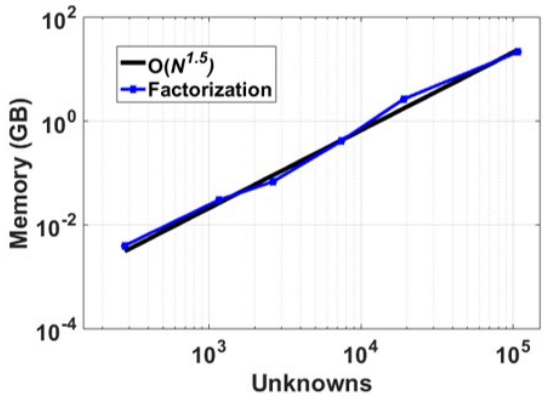

Fig. 6. Factorization memory with an increasing number of unknowns.

IV. NUMERICAL RESULTS

This section demonstrates the accuracy and efficiency of the proposed new fast direct solver method. The fast direct solver is applied on a single-level symmetric half-compressed block matrix computed with ACA compression tolerance of 1e-3 on 128 GB memory and an Intel (Xeon E5-2670) processor system. We have shown the accuracy of bistatic and monostatic RCS.

Efficiency is shown for the factorization and solution time for different geometries compared with open source fast direct solver code 3D EFIE ie3d code from ButterflyPACK [38] H-Matrix solver. All simulations were carried out using a double-precision data type.

A. Bistatic RCS

The accuracy of the proposed fast direct solver is shown for a 4λ sphere bistatic RCS. The bistatic RCS is computed for the plane wave incident at angle $\theta = 0°$ and $\phi = 0°$, with the observation angle varying from $\theta = 0°$ to $180°$ and $\phi = 0°$. The sphere meshes $\lambda/10$ edge length with 82,515 unknowns. The bistatic RCS is computed from the proposed fast direct solver ButterflyPACK and is compared with the Mie Series analytical method.

It is observed from Fig. 7 that the bistatic RCS computed from the fast direct solver matches well with the Mie Series analytical method.

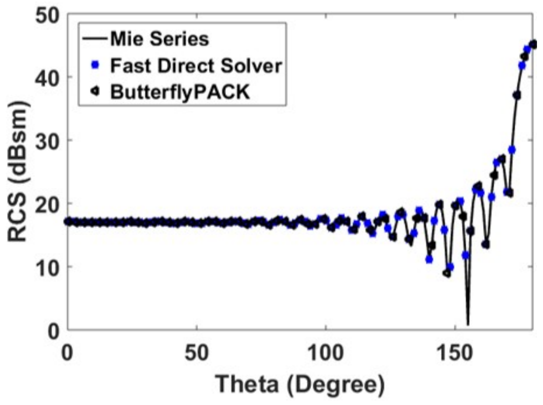

Fig. 7. Bistatic RCS for observation angle $\theta = 0°$ to $180°$ and $\phi = 0°$ with incident angle $\theta = 0°$ and $\phi = 0°$.

Table 1 shows the factorization and solution time and storage memory for the 4λ radius sphere from the proposed fast direct solver and open-source ButterflyPACK. It can be observed that the proposed fast direct solver is 1.3x faster than ButterflyPACK for 4λ sphere RCS computation.

Table 1: Factorization and solution time for 4λ sphere

	Fact. Time (s)	Sol. Time (s)	Total Time (s)	Memory (GB)
ButterflyPACK	26312.73	2.08	26314.81	21
Fast Direct Solver	20129.40	2.65	20132.05	27

B. Monostatic RCS

In this sub-section, we show the accuracy and efficiency of the proposed method for monostatic RCS computation, a multi-RHS problem. The computations are shown for an open and closed structure.

1. Square plate [36]

Monostatic RCS for 181 RHS is computed for the square plate of size $3\lambda \times 3\lambda$ with plane wave incident and observation angle varying from $\theta = -90°$ to $90°$ from $\phi = 270°$. The plate was meshed for $\lambda/10$ element size for 6033 unknowns. The RCS is computed using a ButterflyPACK solver and with the proposed fast direct solver. It is observed from Fig. 8 that the proposed method of RCS computation matches the open-source fast direct solver, and the results are given in [36].

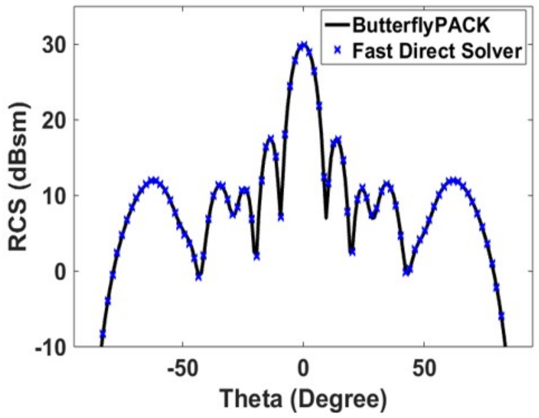

Fig. 8. Monostatic RCS of a square plate for a plane wave incident and observation angle varying from $\theta = -90°$ to $90°$ and $\phi = 270°$.

The factorization and solution time and storage memory in Table 2 are shown for the ButterflyPACK solver and the proposed fast direct solver for 181 RHS.

Table 2: Factorization and solution time for square plate

	Fact. Time (s)	Sol. Time (s)	Total Time (s)	Memory (GB)
ButterflyPACK	26.53	9.49	36.02	0.18
Fast Direct Solver	13.58	7.42	21.02	0.15

Table 2 shows the significant time saving for matrix factorization time and total time with the proposed fast direct solver. The proposed method is 1.7x faster than ButterflyPACK.

2. NASA almond [37]

Monostatic RCS for 180 RHS is computed for the NASA almond with the dimensions as in [35] at 7 GHz with a VV polarized plane wave incident and observation angle varying from $\phi = 0°$ to $180°$ and $\theta° = 90°$. The CAD geometry is meshed for $\lambda/10$ element size for

8761 unknowns. The RCS is computed using the ButterflyPACK solver and with the proposed fast direct solver.

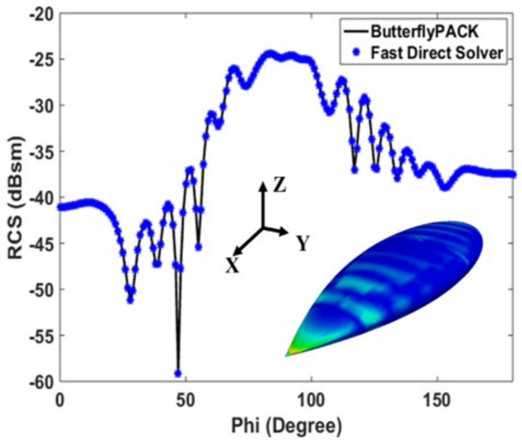

Fig. 9. Monostatic RCS of NASA almond for a plane wave incident and observation angle varying from $\phi =$ $0°$ to $180°$ and $\theta° = 90°$.

It is observed that the proposed method of RCS computation matched with the ButterflyPACK solver, and the results are given in [37].

Table 3: Factorization and solution time for NASA almond

	Fact. Time (s)	Sol. Time (s)	Total Time (s)	Memory (GB)
ButterflyPACK	150.96	18.63	169.59	0.39
Fast Direct Solver	47.32	18.14	65.46	0.44

Matrix factorization and solution time and storage memory in Table 3 are shown for the ButterflyPACK solver and proposed fast direct solver for 180 RHS. The proposed fast direct solver is 2.5x faster than ButterflyPACK.

3. Ship

Monostatic RCS for 180 RHSs is computed for a ship of size 113 m length, 14 m width, and 20 m max height, and is computed at 80 MHz with a VV polarized plane wave. The incident and observation angles vary from $\phi = 0°$ to $180°$ and $\theta° = 90°$. The geometry is meshed for $\lambda/10$ element size for 128,028 unknowns. RCS is computed using a multilevel ButterflyPACK solver and with the proposed fast direct solver.

It is observed from Fig. 10 that the proposed method of RCS computation matched with the ButterflyPACK solver.

The factorization time, solution time, and storage memory in Table 4 is shown for the ButterflyPACK solver and proposed fast direct solver for 180 RHS. It is observed from Table 4 that there is a significant time saving for factorization time and total time with the proposed fast direct solver, and it is 1.3x faster than the ButterflyPACK solver.

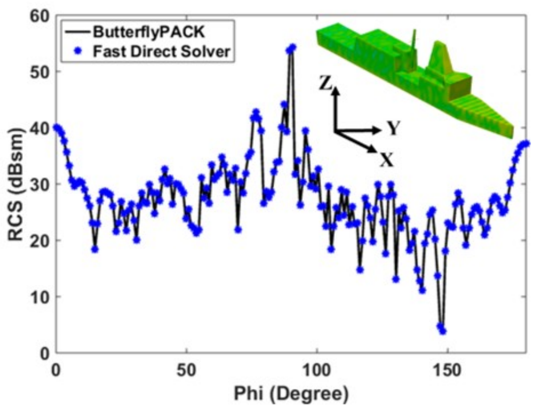

Fig. 10. Monostatic RCS of a ship-like object for a plane wave incident and observation angle varying from $\phi =$ $0°$ to $180°$ and $\theta° = 90°$.

Table 4: Factorization and solution time for the ship

	Fact. Time (s)	Sol. Time (s)	Total Time (s)	Memory (GB)
ButterflyPACK	70694.09	772.30	71466.39	45
Fast Direct Solver	52391.67	1024.91	53416.58	51

V. CONCLUSION

The work proposed in this paper is a new fast direct solver based on the diagonalization method with the guaranteed solution for the MoM linear system of equations. The single-level symmetric half-compressed block matrix is factorized into $\boldsymbol{LDL'}$ where $\boldsymbol{D}$ is a diagonal block matrix, and $\boldsymbol{L'}$ is an upper triangle compressed block matrix. The high cost of diagonalization is overcome using a low-rank block matrix operation process. Our results show the factorization cost scaled to $O(N^{1.5})$. The solution depends on the block diagonal matrix solution process and matrix-vector product of the upper triangle $\boldsymbol{D}$. The far-field compressed matrices in the upper triangle blocks for each row block are merged further to save matrix-vector product cost and storage memory. Also, the solution process time for the proposed factorization scales to $O(N)$. In the current work, the matrix is compressed for the error tolerance of 1e-3. The factorization and solution cost can be reduced by compressing the matrix for lower error tolerance 1e-2. The low cost of the solution process, as demonstrated by illustrative examples, makes it highly desirable for solving multi-RHS problems like monostatic RCS computation or multiport network analysis. Unlike block $\boldsymbol{LU}$ factorization and solution process, which is highly serial in nature, in the proposed method, the block operation can be done independently, making the process amendable for efficient parallelization for both factorization and solution.

REFERENCES

[1] R. F. Harrington, *Field Computation by Moment Methods*. Malabar, Florida: Krieger Publishing Co., 1982.

[2] J.-M. Jin, *The Finite Element Method in Electromagnetics*. Hoboken, New Jersey: John Wiley & Sons, 2015.

[3] A. Taflove and S. C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Norwood, MA: Artech House, 1995.

[4] W. C. Chew, J.-M. Jin, E. Michielssen, and J. Song, *Fast and Efficient Algorithms in Computational Electromagnetics*. Norwood, MA: Artech House, 2000.

[5] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz, "AIM: Adaptive integral method for solving large-scale electromagnetic scattering and radiation problems," *Radio Science*, vol. 31, no. 5, pp. 1225-1251, Dec. 1996.

[6] J. R. Phillips and J. K. White, "A pre-corrected FFT method for electrostatic analysis of complicated 3-D structures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 10, pp. 1059-1072, Oct. 1997.

[7] M. Bebendorf, "Approximation of boundary element matrices," *Numerische Math.*, vol. 86, no. 4, pp. 565-589, June 2000.

[8] S. Kurz, O. Rain, and S. Rjasanow, "The adaptive cross-approximation technique for the 3D boundary-element method," *IEEE Transaction on Magnetics,* vol. 38, no. 2, pp. 421-424, Mar. 2002.

[9] S. Kapur and D. E. Long, "IES3: Efficient electrostatic and electromagnetic solution," *IEEE Computer Science and Engineering*, vol. 5, no. 4, pp. 60-66, Dec. 1998.

[10] W. Hackbusch, "A sparse matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices," *Computing*, vol. 62, no. 2, pp. 89-108, Apr. 1999.

[11] W. Hackbusch and B. N. Khoromskij, "A sparse H-Matrix arithmetic. Part II: Application to multi-dimensional problems," *Computing*, vol. 64, pp. 21-47, Feb. 2000.

[12] Y. K. Negi, "Memory reduced half hierarchal matrix (H-Matrix) for electrodynamic electric field integral equation," *Progress in Electromagnetics Research Letters*, vol. 96, pp. 91-96, Feb. 2021.

[13] A. F. Peterson, "The 'interior resonance' problem associated with surface integral equations of electromagnetics: Numerical consequences and a survey of remedies," *Electromagnetics*, vol. 10, no. 3, pp. 293-312, July 1990.

[14] R. E. Hodges and Y. Rahmat-Samii, "The evaluation of MFIE integrals with the use of vector triangle basis functions," *Microwave and Optical Technology Letters*, vol. 14, no. 1, pp. 9-14, Jan. 1997.

[15] P. Yla-Oijala and M. Taskinen, "Application of combined field integral equation for electromagnetic scattering by dielectric and composite objects," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 3, pp. 1168-1173, Mar. 2005.

[16] Y. Saad, "ILUT: A dual threshold incomplete *LU* factorization," *Numerical Linear Algebra Application*, vol. 1, no. 4, pp. 387-402, 1994.

[17] Y. K. Negi, N. Balakrishnan, Sadasiva M. Rao, and Dipanjan Gope, "Null-field preconditioner with selected far-field contribution for 3-D full-wave EFIE," *IEEE Transactions on Antennas and Propagation*, vol. 64, no. 11, pp. 4923-4928, Aug. 2016.

[18] Y. K. Negi, N. Balakrishnan, and Sadasiva M. Rao, "Symmetric near-field Schur's complement preconditioner for hierarchal electric field integral equation solver," *IET Microwaves, Antennas & Propagation*, vol. 14, no. 14, pp. 1846-1856, Nov. 2020.

[19] D. Gope, I. Chowdhury, and V. Jandhyala, "DiMES: Multilevel fast direct solver based on multipole expansions for parasitic extraction of massively coupled 3D microelectronic structures," in *Proceedings of the 42nd annual Design Automation Conference*, Anaheim, CA, USA, pp. 159-162, June 2005.

[20] D. Sushnikova, L. Greengard, M. O'Neil, and M. Rachh, "FMM-LU: A fast direct solver for multiscale boundary integral equations in three dimensions," *Multiscale Modeling & Simulation*, vol. 21, no. 4, pp. 1570-1601, Dec. 2023.

[21] L. Greengard, D. Gueyffier, P.-G. Martinsson, and V. Rokhlin, "Fast direct solvers for integral equations in complex three-dimensional domains," *Acta Numerica*, vol. 18, pp. 243-275, May 2009.

[22] M. Ma and D. Jiao, "Accuracy directly controlled fast direct solution of general H∧2-Matrices and its application to solving electrodynamic volume integral equations," *IEEE Transactions on Microwave Theory and Techniques*, vol. 66, no. 1, pp. 35-48, Jan. 2018.

[23] Y. K. Negi, N. Balakrishnan, and S. M. Rao, "Fast power series solution of large 3-D electrodynamic integral equation for PEC scatterers," *Applied Computational Electromagnetics Society (ACES) Journal*, vol. 36, no. 10, pp 1301-1311, Oct. 2021.

[24] Y. K. Negi, N. Balakrishnan, and S. M. Rao, "Multilevel power series solution for large surface and volume electric field integral equation," *Applied Computational Electromagnetics Society (ACES) Journal,* vol. 38, no. 5, pp. 297-303, Sep. 2023.

[25] S. Huang and Y. J. Liu, "A new fast direct solver for the boundary element method," *Computational Mechanics*, vol. 60, pp. 379-392, Sep. 2017.

[26] Z. Rong, M. Jiang, Y. Chen, L. Lei, X. Li, and Z. Nie, "Fast direct solution of integral equations with modified HODLR structure for analyzing electromagnetic scattering problems," *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 5, pp. 3288-3296, May 2019.

[27] A. Heldring, J. M. Rius, J. M. Tamayo, J. Parron, and E. Ubeda, "Fast direct solution of Method of Moments linear system," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 11, pp. 3220-3228, Nov. 2007.

[28] J. Shaeffer, "Direct solve of electrically large integral equations for problem sizes to 1 M unknowns," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 8, pp. 2306-2313, Aug. 2008.

[29] W. C. Gibson, "Efficient solution of electromagnetic scattering problems using multilevel adaptive cross approximation and *LU* factorization," *IEEE Transactions on Antennas and Propagation*, vol. 68, no. 5, pp. 3815-3823, May 2020.

[30] S. M. Rao, D. R. Wilton, and A. W. Glisson, "Electromagnetic scattering by surfaces of arbitrary shape," *IEEE Transactions on Antennas and Propagation*, vol. 30, no. 3, pp. 409-418, May 1982.

[31] Y. K. Negi, V. P. Padhy, and N. Balakrishnan, "Re-compressed H-Matrices for fast electric field integral equation," in *IEEE-International Conference on Computational Electromagnetics (ICCEM 2020),* Singapore, pp. 24-26, Aug. 2020.

[32] M. Neumann, "On the Schur complement and the LU-factorization of a matrix," *Linear and Multilinear Algebra*, vol. 9, no. 4, pp. 241-254, Jan. 1981.

[33] S. M. Rao, "A true domain decomposition procedure based on Method of Moments to handle electrically large bodies," *IEEE Transactions on Antennas and Propagation*, vol. 60, no. 9, pp. 4233-4238, Sep. 2012.

[34] Y. K. Negi, N. Balakrishnan, and S. M. Rao, "Schur decomposition fast direct solver for volume surface integral equation," in *2024 IEEE International Symposium on Antennas and Propagation and INC/USNC-URSI Radio Science Meeting (AP-S/INC-USNC-URSI)*, Firenze, Italy, pp. 973-974, July 2024.

[35] J. Shaeffer, "Low-rank matrix algebra for the Method of Moments," *Applied Computational Electromagnetics Society (ACES) Journal*, vol. 33, no. 10, pp. 1052-1059, Oct. 2018.

[36] K. L. Virga and Y. Rahmat-Samii, "RCS characterization of a finite ground plane with perforated apertures: Simulations and measurements," *IEEE Transactions on Antennas and Propagation*, vol. 42, no. 11, pp. 1491-1501, Nov. 1994.

[37] A. C. Woo, H. T. G. Wang, M. J. Schuh, and M. L. Sanders, "EM programmer's notebook-benchmark radar targets for the validation of computational electromagnetics programs," *IEEE Antennas and Propagation Magazine*, vol. 35, no. 1, pp. 84-89, Feb. 1993.

[38] L. Y. Zhuan, *ButterflyPACK* [Online]. Available: https://github.com/liuyangzhuan/ButterflyPACK2

Yoginder Kumar Negi obtained the B.Tech. degree in Electronics and Communication Engineering from Guru Gobind Singh Indraprastha University, New Delhi, India, in 2005, M.Tech. degree in Microwave Electronics from Delhi University, New Delhi, India, in 2007, and the Ph.D. degree in engineering from Indian Institute of Science (IISc), Bangalore, India, in 2018. Negi joined Supercomputer Education Research Center (SERC), IISc Bangalore, in 2008 as a Scientific Officer. He is currently working as a Senior Scientific Officer in SERC IISc Bangalore. His current research interests include numerical electromagnetics, fast techniques for electromagnetic application, bio-electromagnetics, high-performance computing, and antenna design and analysis.

N. Balakrishnan received the B.E. degree (Hons.) in Electronics and Communication from the University of Madras, Chennai, India, in 1972, and the Ph.D. degree from the Indian Institute of Science, Bengaluru, India, in 1979. He joined the Department of Aerospace Engineering, Indian Institute of Science, as an Assistant Professor, in 1981, where he became a Full Professor in 1991, served as Associate Director from 2005 to 2014, and is currently an INSA Senior Scientist at the Supercomputer Education and Research Centre. He has authored over 200 publications in international journals and international conferences. His current research interests include numerical electromagnetics, high-performance computing and networks, polarimetric radars and aerospace electronic systems, information security, and digital library. N. Balakrishnan is a fellow of The World Academy of Sciences (TWAS), the National Academy of Science, the Indian Academy of Sciences, the Indian National Academy of Engineering, the National Academy of Sciences, and the Institution of Electronics and Telecommunication Engineers.

Sadasiva M. Rao obtained his Bachelors, Masters, and Doctoral degrees in electrical engineering from Osmania University, Hyderabad, India, Indian Institute of Science, Bangalore, India, and University of Mississippi, USA, in 1974, 1976, and 1980, respectively. He is well known in the electromagnetic engineering community and included in Thomson Scientifics' *Highly Cited Researchers List*. Rao has been teaching electromagnetic theory, communication systems, electrical circuits, and other related courses at the undergraduate and graduate level for the past 30 years at various institutions. At present, he is working at the Naval Research Laboratories, USA. He has published/presented over 200 papers in various journals/conferences. He is an elected Fellow of IEEE.