IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 31 July 2024, accepted 21 August 2024, date of publication 26 August 2024, date of current version 6 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3449914

RESEARCH ARTICLE

Accelerated Finite Element Method Solver for RCS Analysis Using CUDA-Based Parallel Computing

MINCHEOL JO[1], WOOBIN PARK[1], MOONSEONG KIM[2], AND WOOCHAN LEE[1]
[1]Department of Electrical Engineering, Incheon National University, Incheon 22012, South Korea
[2]Department of IT Convergence Software, Seoul Theological University, Bucheon 14754, South Korea

Corresponding authors: Moonseong Kim (moonseong@stu.ac.kr) and Woochan Lee (wlee@inu.ac.kr)

This work was supported in part by the Laboratory of Computational Electromagnetics for Large-Scale Stealth Platform under Grant UD230016JD, in part by the National Research Foundation of Korea (NRF) grant funded by Korean Government under Grant RS-2023-00242558, and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by Korean Government under Grant 2019-0-00098.

ABSTRACT When addressing large-scale electromagnetic problems using the finite element method (FEM), the resulting matrices are typically sparse, necessitating numerous sparse matrix-vector multiplication (SpMV) operations. To handle this efficiently, research has focused on leveraging large-scale parallel processing with graphics processing units (GPUs). These GPUs can be controlled directly using NVIDIA's Compute Unified Device Architecture (CUDA). In this paper, we analyze electromagnetic scattering for dielectric and dielectric-coated structures using iterative solvers with FEM. To accelerate the handling of the large-scale matrices generated during this process, we employ compressed sparse row (CSR) format, various preconditioners, and CUDA-based GPU parallelization. We verify the accuracy of our results by comparing them with those obtained using the commercial electromagnetic software High Frequency Structure Simulator (HFSS) and our custom-developed MATLAB-based FEM code. Performance improvements are assessed by comparing these results with those from MATLAB's backslash direct solver under single-core processing conditions.

INDEX TERMS Finite element method, absorbing boundary conditions, radar cross section, parallel processing, CUDA.

I. INTRODUCTION

To address electromagnetic phenomena, understanding Maxwell's equations is essential. However, deriving explicit solutions to these equations is notoriously complex. Hence, for practical purposes such as analyzing electromagnetic scattering and antenna radiation, employing electromagnetic numerical analysis becomes imperative. This computational approach enables the approximation of solutions to Maxwell's equations through computer-based methods [1]. One of the most widely used CAD tools for interpreting problems such as antenna and scattering analysis, FEM is a representative electromagnetic numerical analysis technique. It excels in interpreting geometrically complex structures and heterogeneous materials [1], [2], [3], [4]. Also, the system matrix obtained through the finite element process is sparse, offering advantages for iterative methods [3], [5] and suitability for parallel computation with domain decomposition algorithms [6], [7], [8], [9]. Furthermore, Time-domain finite element method (TDFEM) can effectively address nonstationary and nonlinear electromagnetic problems where the properties of the media vary with time [10], [11]. As a result of these advantages, research using TDFEM has been extended, ranging from accelerating TDFEM simulations by extending the time step sizes [11] to developing TDFEM acceleration algorithms for analyzing and designing very large-scale on-chip circuits using the structure of on-chip circuits such as Manhattan geometry and layered permittivity [12].

In electromagnetic problem analysis using the FEM, special boundary conditions such as absorbing boundary conditions (ABC) and perfectly matched layers (PML) are

The associate editor coordinating the review of this manuscript and approving it for publication was Guido Lombardi.

required to truncate the computational domain [13]. ABC is a boundary condition that simulates the behavior of outgoing waves by setting up virtual boundaries at the edges of the computational domain [13], [14]. On the other hand, PML is a boundary condition that minimizes reflections at the boundary by absorbing outgoing waves through the design of an artificial layer [13], [14]. While PML has the advantage of being less influenced by incident angle and frequency, it increases computational time and computer memory due to the additional computational domain corresponding to the artificial layer [13]. Unlike PML, ABC does not require additional artificial layers, which is an advantage. However, it is known that the absorption performance varies depending on the distance between the virtual boundary and the scattering object [13]. It is known that the second order ABC predicts lower reflection compared to the first order ABC. However, it is also known that the finite element matrix with first order ABC provides better conditions than that with second order ABC, making it more efficient for iterative methods [15]. Additionally, according to the previous research conducted by our group, the first-order ABC has been found to perform better compared to PML and waveguide port boundary condition (WPBC) [16].

GPUs have significant advantages in handling computationally intensive data parallel tasks, such as finite element calculations [17]. Therefore, improving FEM speed using GPUs is a major research topic in FEM parallel computing [18]. Recently, GPU hardware and programming strategies have continuously developed [19]. Therefore, analyzing the parallelism of electromagnetic problems and designing efficient GPU parallel algorithms to maximize hardware performance have great potential to address these challenges [20], [21]. Most existing FEM parallel computing studies focus on the two most time-consuming parts solving the system of linear equations [22], [23], [24] and assembling the sparse global matrix [25], [26], [27].

Research has been conducted to accelerate the computation of inverse matrices by developing new algorithms. For example, an algorithm has been proposed that improves the speed of Ridge regression estimator calculations through serial expansion and computational reuse, without adopting inverse matrix computation or other factorization methods [28]. Iterative methods are more suitable for solving large-scale systems of linear equations compared to direct elimination methods, leading to many studies on implementing iterative solvers using parallel computing-based GPUs [5], [29]. The conjugate gradient (CG) method [30], [31], one of the iterative methods, is frequently employed due to its ease of implementation and excellent parallelism [18], [32]. However, directly using the CG method often results in slow convergence because the condition number of the system of linear equations is too large. To achieve better convergence results and improve solution speed, the preconditioned conjugate gradient (PCG) method needs to be used [31], [32]. In large-scale engineering finite element problems, the global matrix is significant, and many of its coefficients are zero. In such problems, the global matrix is usually stored in a sparse matrix format. Therefore, SpMV operations are required during the solution process of the PCG method. The performance of SpMV directly affects the solution time of the PCG method. The GMRES method is used for solving asymmetric and non-Hermitian linear systems and can be parallelized on both CPUs and GPUs. Additionally, recent studies have addressed round-off errors that may arise during the process of generating orthogonal basis vectors [33], [34], [35], [36], [37], [38], [39]. BICG and BICG-STAB require two SpMV operations and several vector-vector operations within each iteration. The efficiency of SpMV depends on the memory access patterns of GPU threads, which are determined by the sparse matrix storage format, making efficient allocation of GPU memory crucial [5]. However, due to the uncertain number of non-zero values in each row of the sparse matrix, irregular memory access patterns may be introduced when designing GPU parallel programs for SpMV [24]. Despite these challenges, some researchers have conducted GPU parallel research on SpMV and achieved excellent results [40], [41], [42], [43], [44], [45], [46], [47].

In this paper, the electromagnetic scattering problems caused by 3D dielectric and dielectric-coated structures in free space are analyzed using FEM to calculate scattered electric field and radar cross section (RCS) parameters. First-order and second-order ABCs are applied to truncate the computational domain, and the accuracy and validity of the simulation results are verified through comparison with the commercial electromagnetic analysis software HFSS, identifying methods with better convergence. Additionally, while previous research applied CUDA C/C++ library-based GPU parallel acceleration for BICG and BICG-STAB methods, this paper implements parallelization of BICG-STAB's SpMV using CUDA C/C++ kernels to directly control GPU operations and compares its performance with the MATLAB backslash direct solver. Furthermore, the efficient compressed sparse row (CSR) format is used for sparse matrix storage, and additional simulation acceleration is performed by applying MATLAB's built-in function "Equilibrate" and SSOR-AI and Jacobi preconditioning techniques. Finally, the performance improvement of GPU parallel computation is compared with single-core CPU computation for each method.

II. FEM WITH BOUNDARY CONDITIONS FORMULATION

In this paper, electromagnetic scattering problems of a 3D dielectric structure in free space, as depicted in Fig. 1, are analyzed. In electromagnetic scattering problems, the electric field is governed by the vector wave equation derived from Maxwell's equations, as shown in (1). To formulate the scattering field, substituting $\mathbf{E} = \mathbf{E}^{sc} + \mathbf{E}^{inc}$ into (1) yields (2) [15].

$$\nabla \times \left[\frac{1}{\mu_r}\nabla \times \mathbf{E}\right] - k_0^2 \varepsilon_r \mathbf{E} = 0 \tag{1}$$

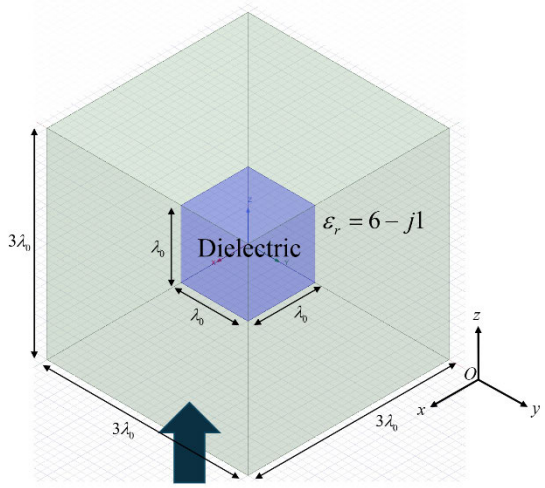


FIGURE 1. 3D dielectric structures in free space.

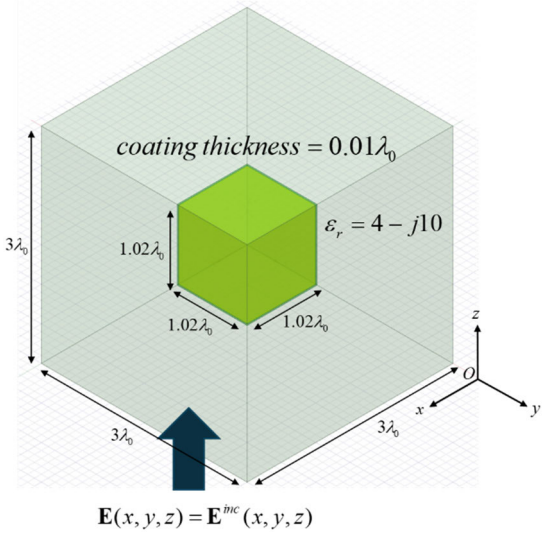


FIGURE 2. 3D dielectric-coated PEC structures in free space.

$$\nabla \times (\frac{1}{\mu_r}\nabla \times \mathbf{E}^{sc}) - k_0^2 \varepsilon_r \mathbf{E}^{sc} = -\nabla \times (\frac{1}{\mu_r}\nabla \times \mathbf{E}^{inc}) + k_0^2 \varepsilon_r \mathbf{E}^{inc} \tag{2}$$

The first and second absorbing boundary conditions can be written as (3) and (4) [15], [48]. Here, $\beta(r)$ is defined as $r/2(1+jk_0r)$, and subscript t and r denote the transverse to $\mathbf{r}$ and radial component, respectively.

$$\mathbf{r} \times (\nabla \times \mathbf{E}^{sc}) \approx -jk_0 \mathbf{r} \times (\mathbf{r} \times \mathbf{E}^{sc}) \tag{3}$$

$$\mathbf{r} \times (\nabla \times \mathbf{E}^{sc}) \approx -jk_0 \mathbf{r} \times (\mathbf{r} \times \mathbf{E}^{sc}) + \beta(r)\nabla \times [\mathbf{r}(\nabla \times \mathbf{E}^{sc})_r] + \beta(r)\nabla_t(\nabla \cdot \mathbf{E}_t^{sc}) \tag{4}$$

Applying the generalized variational principle to (2) and imposing the absorbing boundary condition on an arbitrary surface S to truncate the computational domain yields an equation in the form of functional (5) [15], [48], [49].

$$\begin{aligned} F(\mathbf{E}^{sc}) &= \frac{1}{2}\iiint_V \left[\frac{1}{\mu_r}(\nabla \times \mathbf{E}^{sc}) \cdot (\nabla \times \mathbf{E}^{sc}) - k_0^2 \varepsilon_r \mathbf{E}^{sc} \cdot \mathbf{E}^{sc}\right] dV \\ &+ \iiint_{V_{sc}} \left[\frac{1}{\mu_r}(\nabla \times \mathbf{E}^{sc}) \cdot (\nabla \times \mathbf{E}^{inc}) - k_0^2 \varepsilon_r \mathbf{E}^{sc} \cdot \mathbf{E}^{inc}\right] dV \\ &- \frac{1}{2}\oiint_S \mathbf{E}^{sc} \cdot P(\mathbf{E}^{sc}) dS + \oiint_{S_{sc}} \mathbf{E}^{sc} \cdot (\mathbf{n} \times \nabla \times \mathbf{E}^{inc}) dS \end{aligned} \tag{5}$$

In (5), P is a vector operator, defined as (6) for the first order ABC and (7) for the second order ABC. Additionally, V is the total volume of the computational domain, S is the outer surface of the computational domain, and V_{sc} and S_{sc} are the volume and surface of the scatterer, respectively [15].

$$P(\mathbf{E}^{sc}) = jk_0 \mathbf{r} \times (\mathbf{r} \times \mathbf{E}^{sc}) \tag{6}$$

$$P(\mathbf{E}^{sc}) = jk_0 \mathbf{r} \times (\mathbf{r} \times \mathbf{E}^{sc}) - \beta(r)\nabla \times \left[\mathbf{r}\left(\nabla \times \mathbf{E}^{sc}\right)_r\right] - \beta(r)\nabla_t(\nabla \cdot \mathbf{E}_t^{sc}) \tag{7}$$

Expanding (5) yields the following expression. Additionally, the region corresponding to V uses volume elements(3D vector basis functions $\mathbf{N}^e$), and the region corresponding to S uses surface elements (2D vector basis functions $\mathbf{N}^s$) [15].

$$\begin{aligned} F &= \frac{1}{2}\sum_{e=1}^{V} \{E^e\}^{\mathrm{T}} [K^e] \{E^e\} + \frac{1}{2}\sum_{s=1}^{S} \{E^s\}^{\mathrm{T}} [B^s] \{E^s\}^{\mathrm{T}} \\ &- \sum_{s=1}^{S_{sc}} \{E^s\}^{\mathrm{T}} \{b^s\} \end{aligned} \tag{8}$$

$$\begin{aligned} [K^e] &= \iiint_V \left[\frac{1}{\mu_r}(\nabla \times \mathbf{N}^e) \cdot (\nabla \times \mathbf{N}^e) - k_0^2 \varepsilon_r \mathbf{N}^e \cdot \mathbf{N}^e\right] dV \\ &+ \iiint_{V_{sc}} \left[\frac{2}{\mu_r}(\nabla \times \mathbf{N}^e) \cdot (\nabla \times \mathbf{E}^{inc}) - k_0^2 \varepsilon_r \mathbf{N}^e \cdot \mathbf{E}^{inc}\right] dV \\ [B^s] &= -\oiint_S \mathbf{N}^s \cdot P(\mathbf{N}^s) dS \\ \{b^s\} &= \oiint_{S_{sc}} \mathbf{N}^s \cdot (\mathbf{n} \times \nabla \times \mathbf{E}^{inc}) dS \end{aligned} \tag{9}$$

Furthermore, applying the Ritz procedure yields the following matrix formulation as

$$[K]\{E\} = \{b\} \tag{10}$$

If the analysis is conducted on a perfectly conducting scatterers, (5) can be simplified as

$$F(\mathbf{E}^{sc}) = \frac{1}{2}\iiint_V \left[\frac{1}{\mu_r}(\nabla\times\mathbf{E}^{sc})\cdot(\nabla\times\mathbf{E}^{sc}) - k_0^2\varepsilon_r\mathbf{E}^{sc}\cdot\mathbf{E}^{sc}\right]dV - \frac{1}{2}\oiint_S \mathbf{E}^{sc}\cdot P(\mathbf{E}^{sc})dS \tag{11}$$

In the case of perfectly conducting scatterers excitation is introduced through application of the Dirichlet boundary condition $\mathbf{n}\times\mathbf{E}^{sc} = -\mathbf{n}\times\mathbf{E}^{inc}$. Due to this difference, the efforts required in their numerical implementations are different [15].

However, as shown in the structure of Fig. 2, many metal objects are often coated with a dielectric material for various purposes (such as protection, functionality, etc.). In the case of such structures, it is necessary to consider both aforementioned methods simultaneously. If the dielectric coating is relatively thin compared to the scatterer, the mesh for the dielectric coating becomes relatively small, leading to multi-scale and other issues, making accurate analysis difficult. To address this problem, the impedance boundary condition (IBC) should be applied. This approach has the advantage that the field inside the IBC does not need to be analyzed, regardless of the shape and structure of the object.

$$\begin{aligned}F(\mathbf{E}^{sc}) &= \frac{1}{2}\iiint_V [\frac{1}{\mu_r}(\nabla\times\mathbf{E}^{sc})\cdot(\nabla\times\mathbf{E}^{sc}) - k_0^2\varepsilon_r\mathbf{E}^{sc}\cdot\mathbf{E}^{sc}]dV \\ &+ \frac{1}{2}\oiint_S \mathbf{E}^{sc}\cdot P(\mathbf{E}^{sc})dS \\ &+ \iiint_{V_{sc}} \frac{1}{\mu_r}[(\nabla\times\mathbf{E}^{sc})\cdot(\nabla\times\mathbf{E}^{inc}) - k_0^2\varepsilon_r\mathbf{E}^{sc}\cdot\mathbf{E}^{inc}]dV \\ &- \oiint_{S_{IMP}} \mathbf{E}^{sc}\cdot(\mathbf{n}\times(\nabla\times\mathbf{E}^{inc}))dS \\ &+ \frac{jk_0Z_0}{K}\oiint_{S_{IMP}} (\mathbf{n}\times\mathbf{E}^{sc})\cdot(\mathbf{n}\times\mathbf{E}^{sc})dS \\ &+ \frac{jk_0Z_0}{K}\oiint_{S_{IMP}} (\mathbf{n}\times\mathbf{E}^{sc})\cdot(\mathbf{n}\times\mathbf{E}^{inc})dS\end{aligned} \tag{12}$$

The modified equation for applying the IBC is given by (12) and expanding (12) yields the following expression. Additionally, S_{IMP} is the surface for the impedance boundary (outer surface of the coated dielectric).

$$\begin{aligned}F &= \frac{1}{2}\sum_{e=1}^{V}\{E^e\}^{\mathrm{T}}[K^e]\{E^e\} + \frac{1}{2}\sum_{s=1}^{S}\{E^s\}^{\mathrm{T}}[B_1^s]\{E^s\}^{\mathrm{T}} \\ &+ \frac{1}{2}\sum_{s=1}^{S_{IMP}}\{E^s\}^{\mathrm{T}}[B_2^s]\{E^s\}^{\mathrm{T}} - \sum_{s=1}^{S_{IMP}}\{E^s\}^{\mathrm{T}}\{b^s\}\end{aligned} \tag{13}$$

$$[K^e] = \iiint_V \left[\frac{1}{\mu_r}(\nabla\times\mathbf{N}^e)\cdot(\nabla\times\mathbf{N}^e) - k_0^2\varepsilon_r\mathbf{N}^e\cdot\mathbf{N}^e\right]dV + \iiint_{V_{sc}} \left[\frac{2}{\mu_r}(\nabla\times\mathbf{N}^e)\cdot(\nabla\times\mathbf{E}^{inc}) - k_0^2\varepsilon_r\mathbf{N}^e\cdot\mathbf{E}^{inc}\right]dV$$

$$\begin{aligned}[B^s] &= [B_1^s] + [B_2^s] = -\oiint_S \mathbf{N}^s\cdot P(\mathbf{N}^s)dS \\ &+ 2\frac{jk_0Z_0}{K}\oiint_{S_{IMP}} (\mathbf{n}\times\mathbf{N}^s)\cdot(\mathbf{n}\times\mathbf{N}^s)dS\end{aligned}$$

$$\begin{aligned}\{b^s\} &= \oiint_{S_{sc}} \mathbf{N}^s\cdot(\mathbf{n}\times\nabla\times\mathbf{E}^{inc})dS \\ &- \oiint_{S_{IMP}} \mathbf{N}^s\cdot(\mathbf{n}\times(\nabla\times\mathbf{E}^{inc}))dS \\ &+ \frac{jk_0Z_0}{K}\oiint_{S_{IMP}} (\mathbf{n}\times\mathbf{N}^s)\cdot(\mathbf{n}\times\mathbf{E}^{inc})dS\end{aligned} \tag{14}$$

where $[K]$ is assembled form $[K^e]$ and $[B^s]$, and $\{b\}$ is assembled form $\{b^s\}$. This system can be solved using either direct methods or iterative methods. When solved using iterative methods, it is characterized by the need for a large number of SpMV.

III. RCS ANALYSIS FORMULATIONS

RCS is a measure of how electromagnetic waves are reflected off an object, determined by the magnitude of radar signals scattered from the reflection of emitted electromagnetic waves off the target object. RCS is a critical parameter in typical electromagnetic scattering problems, but it is a complex physical quantity associated with various variables such as the size, shape, and material of the object, as well as factors like radar frequency and incident angle [57].

In 3-D, RCS is defined as (15), where $\mathbf{u}_{far}^{scat}$ represents the scattered electric or magnetic field from a specified direction at the far-zone [57].

$$\sigma_{3D} = \lim_{R\to\infty} 4\pi R^2 \frac{\left|\mathbf{u}_{far}^{scat}\right|^2}{\left|\mathbf{u}^{inc}\right|^2} \tag{15}$$

The far-zone scattered fields can be calculated through the post-processing of FEM. Once the scattered electric field is obtained through FEM, the electric and magnetic fields in the far-zone can be calculated using Huygens' surface equivalence principle as expressed in (16) [57]. $\mathbf{J}$ and $\mathbf{M}$ are the electric current density and magnetic current density, respectively, and S' is Huygens' surface. R is chosen to be sufficiently large to satisfy the far-field assumption.

$$\begin{aligned}\mathbf{E}_{far}^{scat}(\mathbf{r}) &= jk\frac{e^{-jkR}}{4\pi R}\oiint_{s'} \\ &\times\left[\mathbf{r}\times\mathbf{M}(\mathbf{r}') + \eta\mathbf{r}\times(\mathbf{r}\times\mathbf{J}(\mathbf{r}'))\right]e^{-jk(\mathbf{r}'\cdot\mathbf{r})}ds' \\ \mathbf{H}_{far}^{scat}(\mathbf{r}) &= jk\frac{e^{-jkR}}{4\pi R}\oiint_{s'} \\ &\times\left[\mathbf{J}(\mathbf{r}')\times\mathbf{r} + \frac{1}{\eta}\mathbf{r}\times(\mathbf{r}\times\mathbf{M}(\mathbf{r}'))\right]e^{-jk(\mathbf{r}'\cdot\mathbf{r})}ds'\end{aligned} \tag{16}$$

When the outermost surface of the scatterer is a dielectric, the Huygens' surface must be chosen as a closed surface slightly offset from the outermost boundary of the scatterer. In FEM analysis, since the domain is discretized, the closed surface is chosen to be offset by one element. Consequently, the integral over the Huygens' surface in (16) is expressed as the sum of K elements adjacent to the Huygens' surface in (17). In (17), $\mathbf{I}_1$ and $\mathbf{I}_2$ for each coordinate correspond to the equations given in (18) [57].

$$
\begin{aligned}
&(E_{far}^{scat})_x \\
&= jk\frac{e^{-jkR}}{4\pi R}\sum_{i=1}^{K} \\
&\quad \times (I_{1x}^{(i)} + \eta I_{2x}^{(i)}) e^{jk(x_m^{(i)}\sin\theta\cos\varphi + y_m^{(i)}\sin\theta\sin\varphi + z_m^{(i)}\cos\theta)}\Delta s^{(i)} \\
&(E_{far}^{scat})_y \\
&= jk\frac{e^{-jkR}}{4\pi R}\sum_{i=1}^{K} \\
&\quad \times (I_{1y}^{(i)} + \eta I_{2y}^{(i)}) e^{jk(x_m^{(i)}\sin\theta\cos\varphi + y_m^{(i)}\sin\theta\sin\varphi + z_m^{(i)}\cos\theta)}\Delta s^{(i)} \\
&(E_{far}^{scat})_z \\
&= jk\frac{e^{-jkR}}{4\pi R}\sum_{i=1}^{K} \\
&\quad \times (I_{1z}^{(i)} + \eta I_{2z}^{(i)}) e^{jk(x_m^{(i)}\sin\theta\cos\varphi + y_m^{(i)}\sin\theta\sin\varphi + z_m^{(i)}\cos\theta)}\Delta s^{(i)}
\end{aligned}
\tag{17}
$$

$$
\begin{aligned}
&\mathbf{I}_1 \\
&= \mathbf{r}\times\mathbf{M} = \mathbf{r}\times\mathbf{E}\times\mathbf{n} = I_{1x}\mathbf{x} + I_{1y}\mathbf{y} + I_{1z}\mathbf{z} \\
&\mathbf{I}_2 \\
&= \mathbf{r}\times\mathbf{r}\times\mathbf{J} = \mathbf{r}\times\mathbf{r}\times\mathbf{n}\times\mathbf{H} = I_{2x}\mathbf{x} + I_{2y}\mathbf{y} + I_{2z}\mathbf{z}
\end{aligned}
\tag{18}
$$

Ultimately, the RCS obtained through FEM post-processing can be calculated using the far-field values obtained from (17) through the calculation shown in (19) [57].

$$
\sigma_{3D} = 4\pi R^2\left[\left|\left(E_{far}^{scat}\right)_x\right|^2 + \left|\left(E_{far}^{scat}\right)_y\right|^2 + \left|\left(E_{far}^{scat}\right)_z\right|^2\right] \tag{19}
$$

IV. ITERATIVE SOLVER WITH CUDA-BASED PARALLEL COMPUTING

The FEM solver can be solved by direct and iterative methods. It is known that the complexity and memory requirements of direct methods increase exponentially with the dimension of the FEM system matrix. On the other hand, iterative solvers require significantly less memory compared to direct solvers, making them well-suited for large-scale matrix problems [5], [15]. Therefore, in this paper, we apply CUDA-based GPU parallel processing to perform finite element analysis using the more concise and efficient BICG and BICG-STAB methods compared to other iterative methods and compare them with the MATLAB backslash method.

A. CUDA PROGRAM OUTLINE

CUDA is a programming interface provided by NVIDIA, one of the GPU manufacturers, to enable the use of GPUs for general-purpose computing on graphics processing units (GPGPU). CUDA is essentially an extension language of C/C++, allowing modules written in CUDA to be invoked and used from languages such as Fortran, Java, Python, and others. However, to directly control the GPU via CUDA, code must be written in CUDA C/C++. A CUDA program consists of host code and device code, and the typical flow of a CUDA program is depicted in Fig. 2 [16], [58].

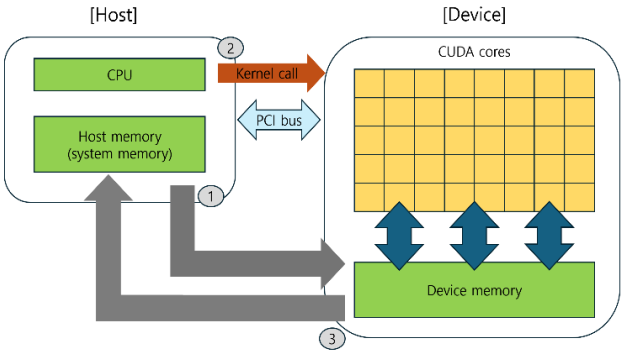


FIGURE 3. Flow of CUDA programs.

Because the CPU and GPU are independent devices, they have separate memory spaces. Therefore, to perform GPU computations, it is necessary to copy the data required for the operations from host memory to device memory. Once the copying process is completed, GPU computation begins via kernel calls, and the computation results are stored in device memory. Subsequently, copying the data stored in device memory back to host memory allows for verification of the computation results [16], [58].

B. CUDA-BASED PARALLEL COMPUTING IMPLEMENTATION

Most iterative algorithms, including BICG and BICG-STAB, require SpMV. In these computations, each element necessitates multiplication and addition, which can be parallelized as they are independent operations. Moreover, finite element matrices are often in sparse matrix, containing many zeros, which reduces the number of elements requiring multiplication. This makes SpMV more efficiently calculated.

In this paper, BICG and BICG-STAB solvers are employed to solve the finite element matrices, as depicted in algorithm 1, 2 [59]. To implement the BICG and BICG-STAB methods, it is necessary to compute SpMV in the form of $y = A\cdot x$, where A is an $m\times n$ matrix, x is a vector of size n, and y is a vector of size m. The calculation for y can be represented as $y_i = \sum_{j=1}^{n} A_{ij}\cdot x_j$ since each row of the vector y is independent, we can utilize parallel processing to compute them simultaneously. Moreover, the parts corresponding to y that can be parallelized in BICG and BICG-STAB are Ap_j, $\hat{p}_j A$, As_j. In BICG and BICG-STAB, there are 2 and 6 parts

corresponding to y, respectively. Therefore, in this paper, CUDA was applied to accelerate computation by directly controlling GPU, large-scale processing devices with a significant number of computational cores, compared to CPU, for SpMV.

Algorithm 1 BICG [59]

Initialize : $x_0 = 0,\ r_0 = b - Ax_0,\ \hat{r}_0 = r_0,\ p_0 = r_0,\ \hat{p}_0 = \hat{r}_0$
Repeat (j = 0, 1, ...)
$\alpha_j \leftarrow \frac{\hat{r}_j r_j}{\hat{p}_j A p_j}$
$x_{j+1} \leftarrow x_j + \alpha_j \cdot p_j$
$\hat{x}_{j+1} \leftarrow \hat{x}_j + \alpha_j \cdot \hat{p}_j$
$r_{j+1} \leftarrow r_j - \alpha_j \cdot \hat{p}_j A$
$\beta_j \leftarrow \frac{\hat{r}_{j+1} r_{j+1}}{\hat{r}_j r_j}$
$p_{j+1} \leftarrow r_{j+1} + \beta_j \cdot p_j$
$\hat{p}_{j+1} \leftarrow \hat{r}_{j+1} + \beta_j \cdot \hat{p}_j$

Algorithm 2 BICG-STAB [59]

Initialize : $x_0 = 0,\ r_0 = b - Ax_0,\ p_0 = r_0$
Repeat (j = 0, 1, ...)
$\alpha_j \leftarrow (r_j, r_0^*)/(Ap_j, r_0^*)$
$s_j \leftarrow r_j - \alpha_j A p_j$
$\omega_j \leftarrow (As_j, s_j)/(As_j, As_j)$
$x_{j+1} \leftarrow x_j + \alpha_j p_j + \omega_j s_j$
$r_{j+1} \leftarrow s_j - \omega_j A s_j$
$\beta_j \leftarrow \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \times \frac{\alpha_j}{\omega_j}$
$p_{j+1} \leftarrow r_{j+1} + \beta_j (p_j - \omega_j A p_j)$

Algorithm 1 involves SpMV computations, such as Ap_j and $\hat{p}_j A$, at each iteration step. To accelerate these computations, information about matrix A is stored in CSR format, and a kernel function is directly developed to parallelize element-wise multiplication and addition. Similarly, Algorithm 2 includes computations such as Ap_j and As_j at each iteration step. Acceleration for Algorithm 2 was performed in the same manner as Algorithm 1.

V. SIMULATION RESULTS

A. TESTING AND VALIDATION OF AN IN-HOUSE FEM CODE

In this paper, we analyze the electric field and RCS parameters for the dielectric structure shown in Fig. 1 and the dielectric coating structure shown in Fig. 2. In this analysis, the incident waves from arbitrary directions are given by (20) and (21), respectively.

$$\mathbf{E}^{inc} = \frac{1}{\sqrt{13}}(\sqrt{3}\mathbf{x} + \mathbf{y} - 3\mathbf{z})e^{-jk_0(\sqrt{3}x+3y+2z)/4} \tag{20}$$

$$\mathbf{E}^{inc} = \mathbf{y}e^{-jk_0 z} \tag{21}$$

The operating frequency is set to 300 MHz, corresponding to a wavelength of 1 meter. The dielectric box is a cube with each side equal to one wavelength, and its permittivity is 6-j1. Additionally, the arbitrary boundary surface is enclosed by a cube with each side three wavelengths. Therefore, the distance from the surface of the dielectric to the arbitrary boundary surface is one wavelength. In the case of the dielectric-coated box, it consists of a PEC cube with each side equal to one wavelength, coated with a dielectric layer of 0.01 wavelength thickness and a permittivity of 4-j10. The arbitrary boundary surface is similarly enclosed by a cube with each side three wavelengths. Thus, the distance from the dielectric coating surface to the arbitrary boundary surface is 0.99 wavelengths. Additionally, a complete program is developed using MATLAB (compatible with version R2022b). The accuracy of the simulation is verified by comparing the results from the in-house FEM program with those from the commercial electromagnetic software HFSS for the scattered electric field distribution and RCS.

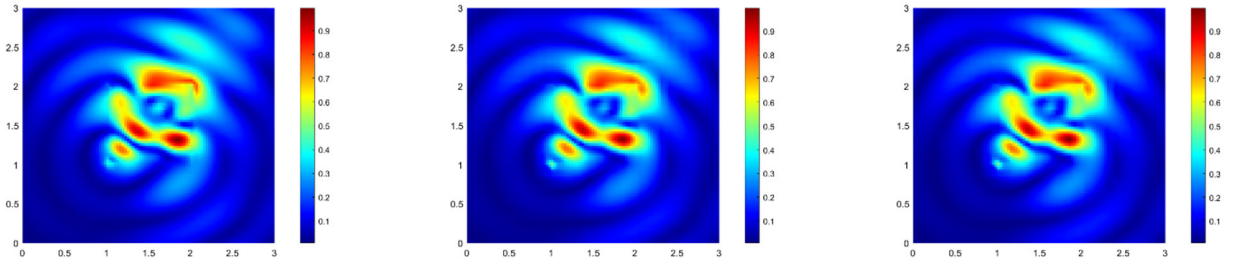
In both cases, the Huygens's surface is defined as a region 0.05 meters away from the scatterer, and the RCS parameters are calculated using the electric current density and magnetic current density passing through this surface. The simulation employs a 3-D edge-based brick element, with a total of 216,000 elements and 669,780 edges used for both first-order and second-order ABCs. The scattered electric field distributions and RCS results for the dielectric structure in the XY, XZ, and YZ planes are shown in Figs. 4 to 7, while the corresponding results for the dielectric-coated structure are shown in Figs. 8 to 11. The simulation results indicate a high level of agreement between HFSS and both first-order and second-order ABCs.

B. COMPARISON OF THE CONVERGENCE CHARACTERISTICS

The BICG-STAB method, based on Krylov subspace techniques, is known for faster and more stable convergence compared to BICG because it does not require the multiplication of the system matrix with its transpose [60]. To determine if this characteristic applies to matrices generated by FEM, convergence simulations based on absorbing boundaries are conducted. In these simulations, the residual for BICG represents the norm of the direction vector **r**, while for BICG-STAB, it represents the smaller value between the norms of the two direction vectors **s** and **r**. Accordingly, the results shown in Figure 12 indicate that BICG exhibits highly unstable convergence for both dielectric and dielectric-coated structures, whereas BICG-STAB shows relatively improved convergence characteristics. Furthermore, it is observed that second-order ABC introduces much larger residuals due to the inclusion of more complex terms compared to first-order ABC. Consequently, the combination of BICG-STAB and first-order ABC achieves the best performance.

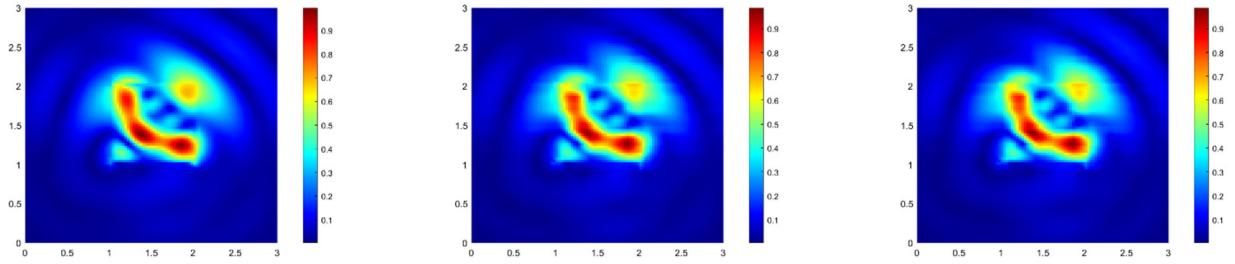
VI. ACCELERATING FEM SOLVER WITH EQUILIBRATE FUNCTION, JACOBI, SSOR-AI PRECONDITIONER

To enhance simulation speed, CUDA-based GPU parallel acceleration is applied to a BICG-STAB based FEM solver



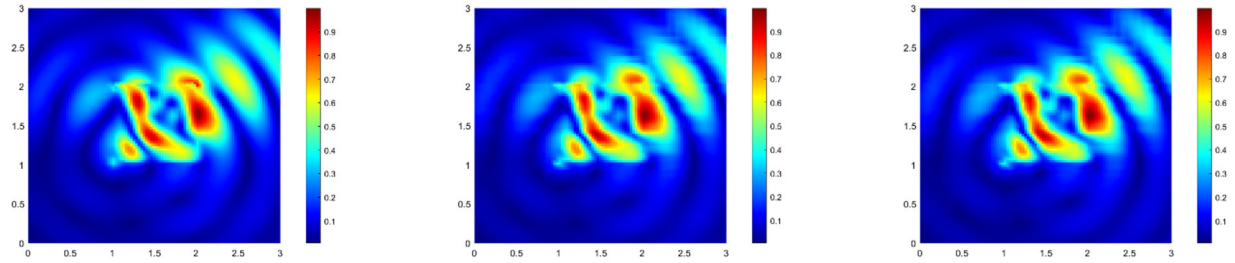
(a) HFSS (b) First order ABC (c) Second order ABC

FIGURE 4. Scattered electric field distribution for a dielectric structure (XY plane).



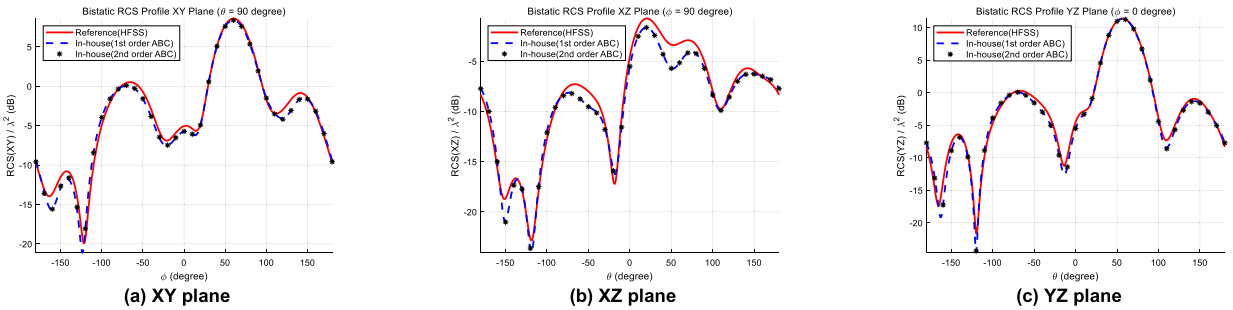
(a) HFSS (b) First order ABC (c) Second order ABC

FIGURE 5. Scattered electric field distribution for a dielectric structure (XZ plane).



(a) HFSS (b) First order ABC (c) Second order ABC

FIGURE 6. Scattered electric field distribution for a dielectric structure (YZ plane).



(a) XY plane (b) XZ plane (c) YZ plane

FIGURE 7. The RCS simulation results for a dielectric structure (each plane).

using the first-order ABC, which has demonstrated the fastest convergence in dielectric and dielectric-coated structures. As outlined in Algorithm 2, the SpMV operations in BICG-STAB occur six times in total, all of which can be accelerated by the GPU. Additionally, to compare convergence speeds, further experiments are conducted by applying the equilibrate function along with Jacobi and SSOR-AI preconditioners.

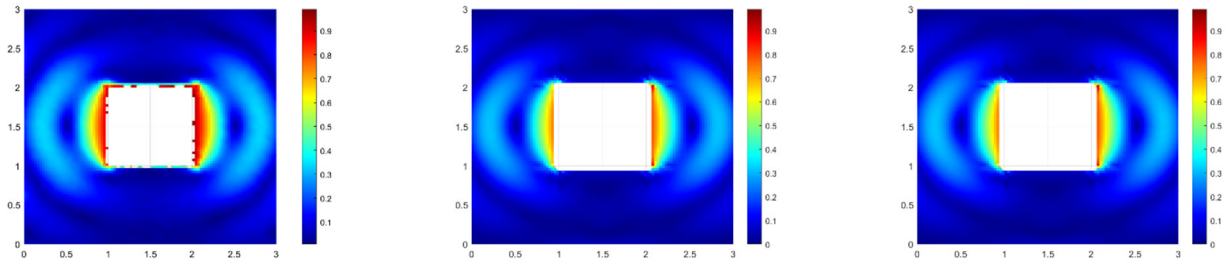


FIGURE 8. Scattered electric field distribution for a dielectric-coated structure (XY plane).

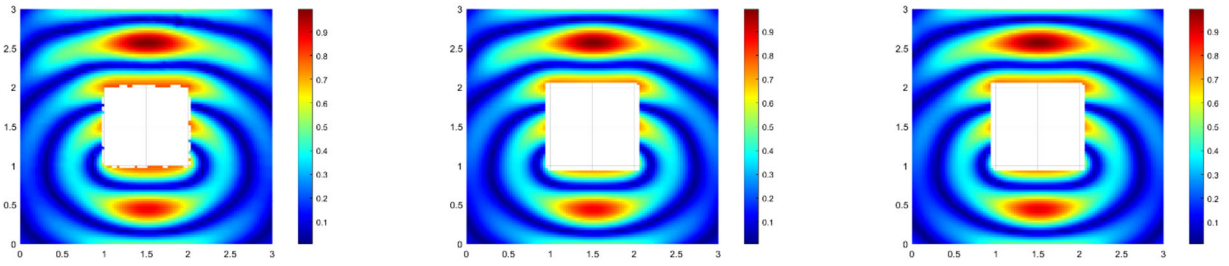


FIGURE 9. Scattered electric field distribution for a dielectric-coated structure (XZ plane).

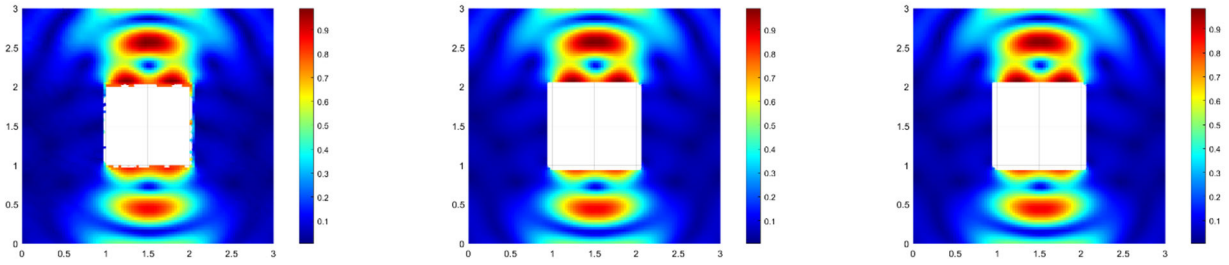


FIGURE 10. Scattered electric field distribution for a dielectric-coated structure (YZ plane).

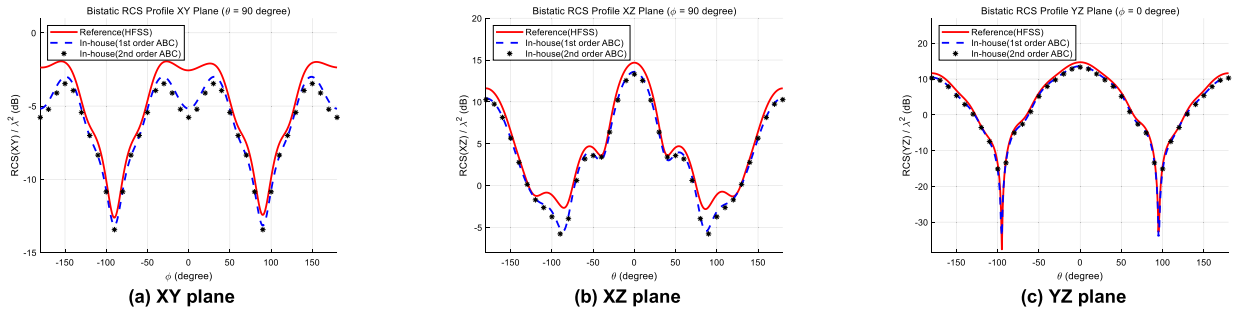
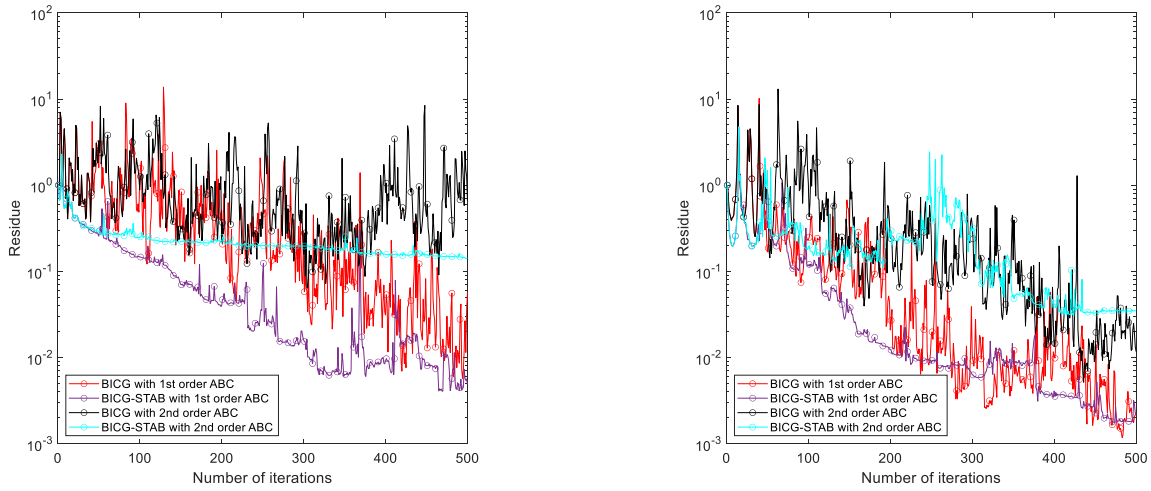


FIGURE 11. The RCS simulation results for a dielectric-coated structure (each plane).

A. MATLAB'S BUILT-IN EQUILIBRATE FUNCTION

When using the function [P, R, C] = equilibrate(A) in MATLAB, matrix A can be rescaled to have unit-size diagonal elements, with off-diagonal entries not exceeding a magnitude of 1, producing the matrix $B = R \times P \times A \times C$. Here, P represents a permutation matrix used to balance the matrix,



(a) Dielectric structure

(b) Dielectric-coated structure

FIGURE 12. Comparing convergence characteristics with varying ABCs and iterative method.

where the absolute product of the diagonal elements of $P \times A$ is maximized by rearranging A. Additionally, R and C denote the row and column diagonal scaling matrices, respectively, aiding in maintaining balance by adjusting the sizes of rows and columns. Additionally, the resulting matrix B the equilibrate function generally possesses a lower condition number than A, which results in enhanced efficiency and stability in the solution of linear systems.

B. SSOR-AI PRECONDITIONER

The SSOR-AI preconditioner is commonly used to accelerate the convergence of iterative solvers for sparse linear systems [5], [42]. Assuming that matrix A is decomposed as shown in (22), the SSOR preconditioner is defined as (23) [42]. Here, D and L respectively represent the diagonal matrix and the lower triangular part of matrix A.

$$A = L + D + L^T \tag{22}$$

$$M = KK^T$$

where

$$K = \frac{1}{\sqrt{2-\omega}} \left(\overline{D} + L\right) \overline{D}^{-1/2} \tag{23}$$

ω represents the relaxation parameter, ranging from 0 to 2. Additionally, $\bar{D}$ is defined as $(1/\omega)D$.

Computing the SSOR preconditioner directly, as in (23), involves substantial computational cost. Conversely, employing the SSOR-AI preconditioner offers a straightforward computational approach [42].

$$\begin{aligned} K^{-1} &= \sqrt{2-\omega}\overline{D}^{1/2}(I + \overline{D}^{-1}L)^{-1}\overline{D}^{-1} \\ &\approx \sqrt{2-\omega}\overline{D}^{1/2} \\ &\quad \times \left[I - \overline{D}^{-1}L + (\overline{D}^{-1}L)^2 - (\overline{D}^{-1}L)^3 + \cdots\right]\overline{D}^{-1} \end{aligned} \tag{24}$$

Applying the Neumann series to (23) transforms it into (24). The first-order approximation inverse of K can be computed as shown in (25), and the first order SSOR-AI preconditioner can be expressed as in (26) [42].

$$K^{-1} \approx \sqrt{2-\omega}\overline{D}^{1/2}(I - \overline{D}^{-1}L)\overline{D}^{-1} = \overline{K} \tag{25}$$

$$\overline{M} = \overline{K}^T\overline{K} \tag{26}$$

TABLE 1. Simulation environment.

Component	Specification
Processor	11th Gen Intel(R) Core(TM) i9-11900 K @ 3.50GHz
RAM	128GB
Operating System	Windows 10 Pro 64bits
Graphics Card	NVIDIA GeForce RTX 3080

To determine the optimal value of ω, simulations are conducted by varying ω and applying the SSOR-AI preconditioner to the structures depicted in Fig. 1 and Fig. 2. The system matrix has a size of 669,780 $\times$ 669,780, and the convergence criterion is set to ensure the residual is below 0.01. Additionally, Fig. 13 presents a graph comparing the number of iterations required for convergence as ω varies. For both dielectric and dielectric-coated structures, convergence is achieved with the minimum number of iterations when $\omega = 0.4$

C. RCS ANALYSIS WITH ACCELERATED FEM SOLVER

In this simulation, RCS analysis and acceleration through CUDA-based GPU parallel processing are performed for the structures shown in Fig. 1 and Fig. 2. Due to the large-scale matrix operations involving matrices of size 669,780 $\times$ 669,780, the combination of BICG-STAB and first-order ABC, which showed the fastest convergence, is used for accelerating these operations. The accuracy of BICG-STAB

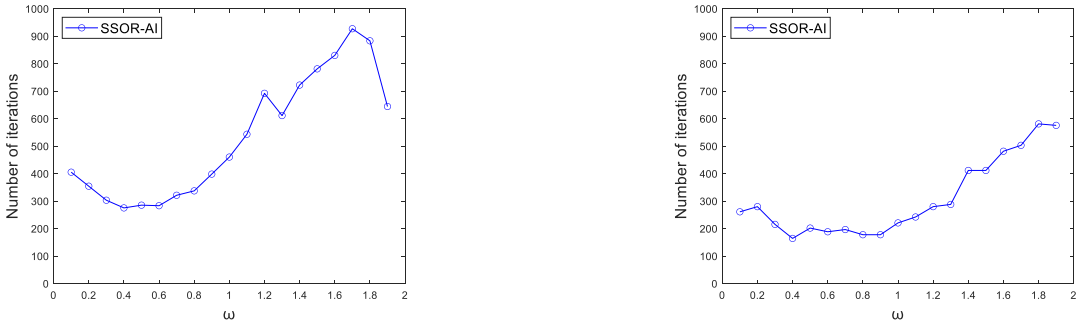


FIGURE 13. The number of iterations with SSOR-AI preconditioner as a function of ω.

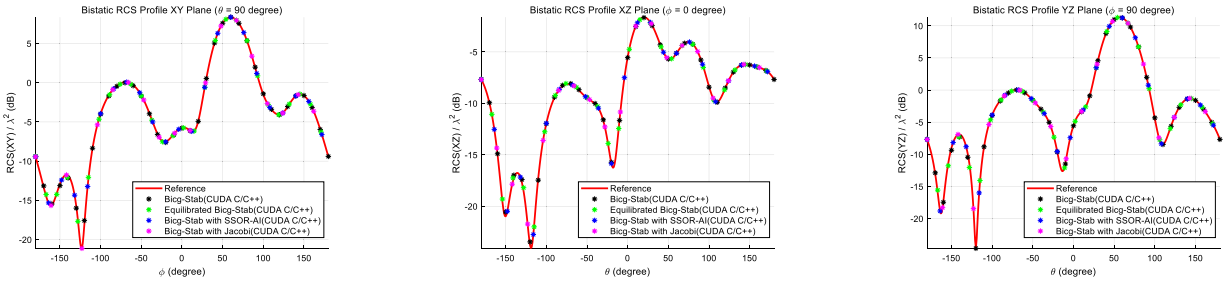


FIGURE 14. RCS simulation results for each plane with different solution techniques (dielectric structure).

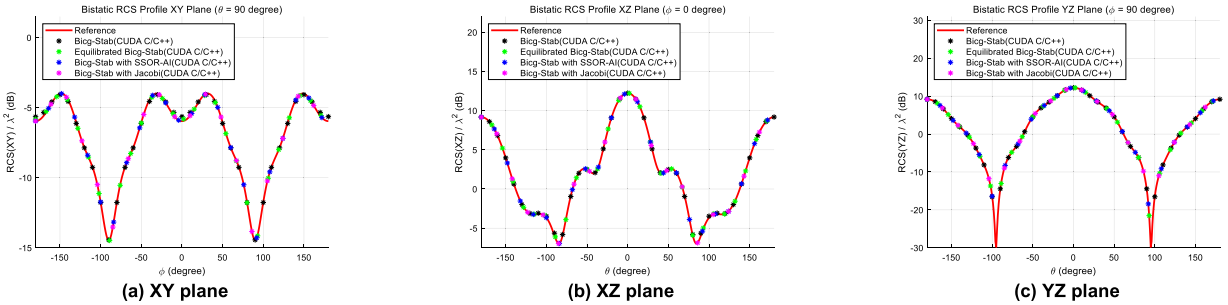
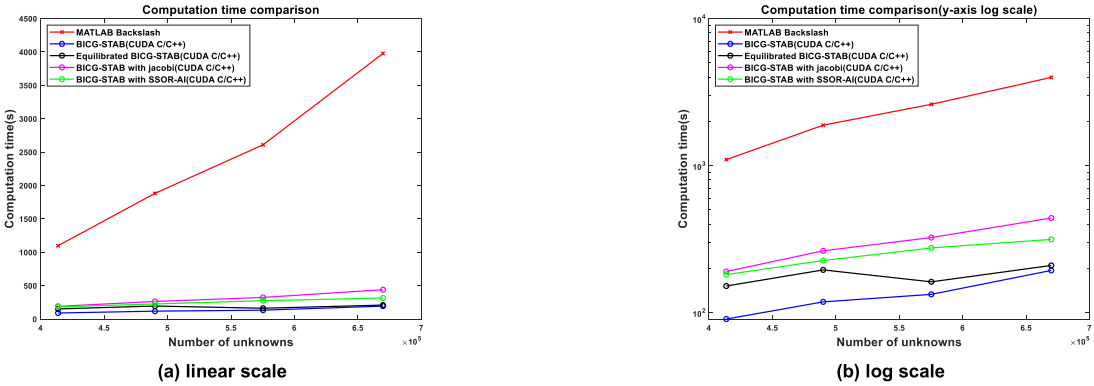


FIGURE 15. RCS simulation results for each plane with different solution techniques (dielectric-coated structure).

and BICG-STAB with preconditioners (equilibrate function, Jacobi, SSOR-AI) is verified through RCS comparison. The simulation environment is as shown in Table 1, with the number of edges being 669,780. Additionally, the tolerance is set to 0.01, and the ω value for SSOR-AI is set to 0.4, which showed the fastest convergence. Figs. 14 and 15 show the RCS results for dielectric and dielectric-coated structures, respectively. Additionally, the results of the BICG-STAB with preconditioners and GPU parallel processing are compared with the MATLAB backslash direct solver using a single CPU core. It can be observed that the results from the iterative solver match those from the direct solver.

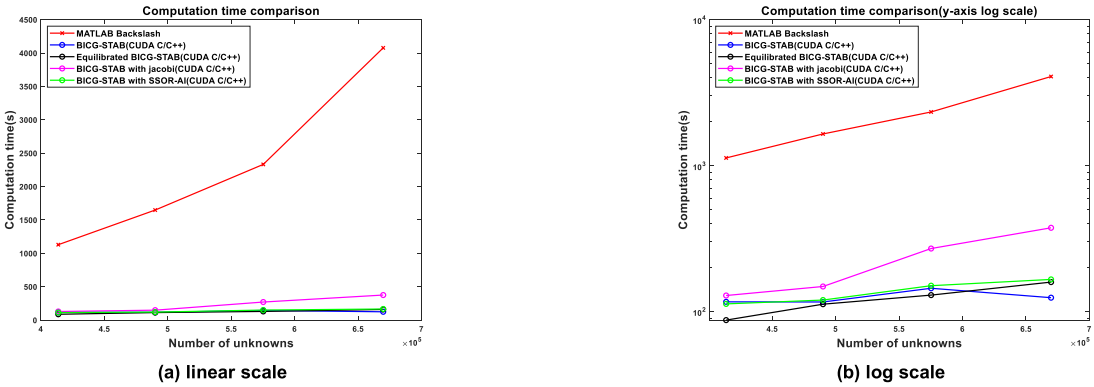
To analyze the performance improvement based on matrix size during CUDA-based GPU parallelization for dielectric and dielectric-coated structures, the number of edges is gradually increased to 413,712, 490,050, 575,244, and 669,780. The computational speed and the number of iterations required for convergence for each method are compared, with the results presented in Figs. 16 and 17 and Tables 2 and 3. Figs. 16 and 17 show the computational speed graphs for each method for dielectric and dielectric-coated structures, respectively, plotted on both linear and logarithmic scales. Table 2 provides detailed analysis results for the dielectric structure, including computation time, speed-up (the ratio of improved speed compared to the direct solver), iteration number, and residual values for each method. Table 3 presents results for the dielectric-coated structure, with detailed items identical to those in Table 2.

According to the simulation results, overall, the BICG-STAB with CUDA-based GPU acceleration shows improved computation time compared to MATLAB backslash. Additionally, as the number of edges, or the number of unknowns,



(a) linear scale (b) log scale

FIGURE 16. Computation time comparison (dielectric structure).



(a) linear scale (b) log scale

FIGURE 17. Computation time comparison (dielectric-coated structure).

TABLE 2. Computation time for solution techniques based on the number of unknowns (dielectric structure).

Solver \ # of unknowns	-	413,712	490,050	575,244	669,780
MATLAB Backslash	-	**1097.92s**	**1882.13s**	**2609.52s**	**3977.82s**
BICG-STAB	**Computation time / Speed up**	**90.28s / 12.16**	**118.15s / 15.93**	**132.89s / 19.63**	**193.31s / 20.57**
	Iteration number / Residual	248 / 0.009548	277 / 0.009302	267 / 0.009895	316 / 0.008738
Equilibrated BICGS-TAB	**Computation time / Speed up**	**151.43s / 7.25**	**194.91s / 9.65**	**161.71s / 16.13**	**208.96s / 19.03**
	Iteration number / Residual	412 / 0.008196	458 / 0.006848	324 / 0.009592	355 / 0.009566
BICG-STAB with Jacobi preconditioner	**Computation time / Speed up**	**190.09s / 5.77**	**262.83s / 7.16**	**323.61s / 8.06**	**439.46s / 9.05**
	Iteration number / Residual	513 / 0.009689	584 / 0.009582	645 / 0.007228	718 / 0.009711
BICG-STAB with SSOR-AI preconditioner	**Computation time / Speed up**	**181.21s / 6.05**	**225.71s / 8.33**	**275.06s / 9.48**	**314.35s / 12.65**
	Iteration number / Residual	263 / 0.009166	272 / 0.009522	274 / 0.00999	275 / 0.009978

increases, the ratio of improved computational speed also increases. This is because the numerous cores of the GPU can be efficiently utilized for parallel processing, fully leveraging the advantages of parallelism. Furthermore, the dielectric-coated structure shows relatively faster computation time compared to the dielectric structure, likely due to the improved matrix properties resulting from the full reflection of the dielectric effect in the IBC. However, none of the preconditioners applied to improve the convergence performance of BICG-STAB, including the equilibrate function, Jacobi, and SSOR-AI, have enhanced the computational speed. Although the SSOR-AI preconditioner has shown

TABLE 3. Computation time for solution techniques based on the number of unknowns (dielectric-coated structure).

# of unknowns / Solver	-	413,712	490,050	575,244	669,780
MATLAB Backslash	-	**1129.32s**	**1648.15s**	**2331.94s**	**4077.27s**
BICG-STAB	**Computation time / Speed up**	**116.73s / 9.67**	**116.49s / 14.14**	**139.81s / 16.12**	**124.65s / 32.71**
	Iteration number / Residual	336 / 0.009685	291 / 0.009130	292 / 0.009875	224 / 0.009359
Equilibrated BICG-STAB	**Computation time / Speed up**	**87.53s / 12.90**	**112.50s / 14.65**	**129.83s / 17.96**	**159.40s / 25.57**
	Iteration number / Residual	240 / 0.007215	267 / 0.009856	269 / 0.009912	289 / 0.009938
BICG-STAB with Jacobi preconditioner	**Computation time / Speed up**	**128.99s / 8.75**	**148.72s / 11.08**	**270.62s / 8.61**	**375.28s / 10.86**
	Iteration number / Residual	360 / 0.009951	365 / 0.009937	562 / 0.007968	662 / 0.009837
BICG-STAB with SSOR-AI preconditioner	**Computation time / Speed up**	**113.23s / 9.99**	**120.10s / 13.72**	**150.58s / 15.48**	**166.26s / 24.52**
	Iteration number / Residual	186 / 0.009014	169 / 0.009965	181 / 0.009959	164 / 0.009963

overall improved convergence iterations, the total computational speed has been relatively slower due to the increased number of nonzeros resulting from the application of SSOR-AI. The lack of improvement in convergence speed despite the application of these preconditioning techniques is likely due to the characteristics of the system matrix generated in frequency domain FEM. This matrix includes both positive and negative eigenvalues, exhibits indefinite properties with complex values, and typically has a large condition number because the smallest eigenvalue is close to zero.

VII. CONCLUSION

In this paper, the electromagnetic scattering problem is analyzed for a three-dimensional cubic dielectric and dielectric-coated structure when an arbitrary electromagnetic wave is incident. The scattered electric field and RCS are compared between the results from the commercial electromagnetic software HFSS and the self-written in-house MATLAB-based FEM code to validate the accuracy of the developed code. As a result, the accuracy of the near-field (electric field) and far-field (RCS) for HFSS and first-order and second-order ABCs is confirmed. Subsequently, the convergence of BICG and BICG-STAB for first-order and second-order ABCs is compared, and it is confirmed that the use of first-order ABC with BICG-STAB provides the most stable convergence.

GPU parallelization are applied for efficient large-scale matrix analysis. For the SSOR-AI preconditioner, the relaxation parameter ω is varied to determine the optimal value of $\omega = 0.4$. Additionally, the RCS results of the basic BICG-STAB and the BICG-STAB with preconditioners are compared with the direct solver MATLAB backslash, confirming high accuracy. Furthermore, the computational performance based on matrix size is evaluated with CUDA-based GPU parallelization, showing that the speed up ratio increases as the matrix size grows. The SSOR-AI preconditioner results in the fewest iterations for convergence, but the basic BICG-STAB without any preconditioner exhibits the fastest computation speed, achieving up to a 32-fold speed up.

In this paper, a large-scale problem is addressed by incrementally increasing the number of edges for a simple structure, which can be extended to the analysis of large and complex structures, such as aircraft carriers, in the future. Additionally, the parallel processing algorithm performed in this study was executed on a desktop specification computer, indicating that even faster analysis will be possible in the future using cluster computing environments that support more processors.

ACKNOWLEDGMENT

(Mincheol Jo and Woobin Park are co-first authors.)

REFERENCES

[1] D. B. Davidson, *Computational Electromagnetics for RF and Microwave Engineering*, 2nd ed., Cambridge, U.K.: Cambridge Univ. Press, 2014, pp. 1–29.

[2] W. B. Park, M. S. Kim, and W. C. Lee, "Absorbing boundary conditions and parallelization for waveguide electromagnetic analysis using finite element method," *J. Internet Comput. Service*, vol. 23, no. 3, pp. 67–76, Jun. 2022.

[3] W. Lee, W. Park, J. Park, Y.-J. Kim, and M. Kim, "Parallel iterative FEM solver with initial guess for frequency domain electromagnetic analysis," *Intell. Autom. Soft Comput.*, vol. 36, no. 2, pp. 1585–1602, Jan. 2023, doi: 10.32604/iasc.2023.033112.

[4] W. B. Park, M. S. Kim, and W. C. Lee, "Commercial and in-house simulator development trend for electromagnetic analysis of autonomous driving environments," *J. Korean Soc. Digit. Ind. Inf. Manag.*, vol. 17, no. 4, pp. 31–42, Dec. 2021, doi: 10.17662/ksdim.2021.17.4.031.

[5] H.-T. Meng, B.-L. Nie, S. Wong, C. Macon, and J.-M. Jin, "GPU accelerated finite-element computation for electromagnetic analysis," *IEEE Antennas Propag. Mag.*, vol. 56, no. 2, pp. 39–62, Apr. 2014, doi: 10.1109/MAP.2014.6837065.

[6] Z. Lou and J.-M. Jin, "A novel dual-field time-domain finite-element domain-decomposition method for computational electromagnetics," *IEEE Trans. Antennas Propag.*, vol. 54, no. 6, pp. 1850–1862, Jun. 2006, doi: 10.1109/TAP.2006.875922.

[7] S. Sun and D. Jiao, "Split-field domain decomposition parallel algorithm with fast convergence for electromagnetic analysis," *IEEE J. Multiscale Multiphys. Comput. Techn.*, vol. 8, pp. 135–146, 2023, doi: 10.1109/JMMCT.2023.3236645.

[8] Y. Li and J.-M. Jin, "A vector dual-primal finite element tearing and interconnecting method for solving 3-D large-scale electromagnetic problems," *IEEE Trans. Antennas Propag.*, vol. 54, no. 10, pp. 3000–3009, Oct. 2006, doi: 10.1109/TAP.2006.882191.

[9] S. Wang, Y. Shao, and Z. Peng, "A parallel-in-space-and-time method for transient electromagnetic problems," *IEEE Trans. Antennas Propag.*, vol. 67, no. 6, pp. 3961–3973, Jun. 2019, doi: 10.1109/TAP.2019.2909937.

[10] Q. Ren, S. Yan, and A. Z. Elsherbeni, *Advances in Time-Domain Computational Electromagnetic Methods*. Hoboken, NJ, USA: Wiley-IEEE Press, 2022, pp. 81–128.

[11] W. Lee and D. Jiao, "An alternative explicit and unconditionally stable time-domain finite-element method for electromagnetic analysis," *IEEE J. Multiscale Multiphys. Comput. Techn.*, vol. 3, pp. 16–28, 2018, doi: 10.1109/JMMCT.2018.2814480.

[12] W. Lee and D. Jiao, "Fast structure-aware direct time-domain finite-element solver for the analysis of large-scale on-chip circuits," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 5, no. 10, pp. 1477–1487, Oct. 2015, doi: 10.1109/TCPMT.2015.2472403.

[13] M. C. Jo, W. B. Park, and W. C. Lee, "Comparison of absorbing boundary conditions and waveguide port boundary condition for waveguide electromagnetic analysis using finite element method," *J. Internet Comput. Services*, vol. 24, no. 2, pp. 27–36, Apr. 2023, doi: 10.7472/jksii.2023.24.2.27.

[14] D. Jiao and J.-M. Jin, "An effective algorithm for implementing perfectly matched layers in time-domain finite-element simulation of open-region EM problems," *IEEE Trans. Antennas Propag.*, vol. 50, no. 11, pp. 1615–1623, Nov. 2002, doi: 10.1109/TAP.2002.803987.

[15] J. M. Jin, *The Finite Element Method in Electromagnetics*, 3rd ed., Hoboken, NJ, USA: Wiley-IEEE Press, 2013, pp. 315–338.

[16] W. Park, M. Jo, M. Kim, and W. Lee, "Boundary conditions comparison for electromagnetic simulation using the finite element method with CUDA computing," *J. Electr. Eng. Technol.*, pp. 1–10, Mar. 2024, doi: 10.1007/s42835-024-01887-8.

[17] NVIDIA. (Jul. 2024). *NVIDA C++ Programming Guide Release*. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[18] N. K. Pikle, S. R. Sathe, and A. Y. Vyavhare, "GPGPU-based parallel computing applied in the FEM using the conjugate gradient algorithm: A review," *Sādhanā*, vol. 43, no. 7, p. 111, Jun. 2018, doi: 10.1007/s12046-018-0892-0.

[19] A. R. Brodtkorb, T. R. Hagen, and M. L. Sætra, "Graphics processing unit (GPU) programming strategies and trends in GPU computing," *J. Parallel Distrib. Comput.*, vol. 73, no. 1, pp. 4–13, Jan. 2013, doi: 10.1016/j.jpdc.2012.04.003.

[20] S. Georgescu, P. Chow, and H. Okuda, "GPU acceleration for FEM-based structural analysis," *Arch. Comput. Methods Eng.*, vol. 20, no. 2, pp. 111–121, Apr. 2013, doi: 10.1007/s11831-013-9082-8.

[21] Y. K. Gujjala, H.-M. Kim, and D.-W. Ryu, "GPGPU-based parallel computation using discrete elements in geotechnics: A state-of-art review," *Arch. Comput. Methods Eng.*, vol. 30, no. 3, pp. 1601–1622, Apr. 2023, doi: 10.1007/s11831-022-09851-3.

[22] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder, "Sparse matrix solvers on the GPU: Conjugate gradients and multigrid," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 917–924, Jul. 2003, doi: 10.1145/882262.882364.

[23] J. Martínez-Frutos, P. J. Martínez-Castejón, and D. Herrero-Pérez, "Fine-grained GPU implementation of assembly-free iterative solver for finite element problems," *Comput. Struct.*, vol. 157, pp. 9–18, Sep. 2015, doi: 10.1016/j.compstruc.2015.05.010.

[24] N. K. Pikle, S. R. Sathe, and A. Y. Vyavahare, "Low occupancy high performance elemental products in assembly free FEM on GPU," *Eng. Comput.*, vol. 38, no. 3, pp. 2189–2204, Aug. 2022, doi: 10.1007/s00366-021-01350-6.

[25] C. Cecka, A. J. Lew, and E. Darve, "Assembly of finite element methods on graphics processors," *Int. J. Numer. Methods Eng.*, vol. 85, no. 5, pp. 640–669, Feb. 2011, doi: 10.1002/nme.2989.

[26] U. Kiran, D. Sharma, and S. S. Gautam, "GPU-warp based finite element matrices generation and assembly using coloring method," *J. Comput. Des. Eng.*, vol. 6, no. 4, pp. 705–718, Oct. 2019, doi: 10.1016/j.jcde.2018.11.001.

[27] R. Zayer, M. Steinberger, and H.-P. Seidel, "Sparse matrix assembly on the GPU through multiplication patterns," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2017, pp. 1–8, doi: 10.1109/HPEC.2017.8091057.

[28] W. C. Lee, M. S. Kim, and J. Y. Park, "Speed-up of the matrix computation on the ridge regression," *KSII Trans. Internet Inf. Syst.*, vol. 15, no. 10, pp. 3482–3497, Oct. 2021, doi: 10.3837/tiis.2021.10.003.

[29] A. F. P. de Camargos and V. C. Silva, "Performance analysis of multi-GPU implementations of Krylov-subspace methods applied to FEA of electromagnetic phenomena," *IEEE Trans. Magn.*, vol. 51, no. 3, pp. 1–4, Mar. 2015, doi: 10.1109/TMAG.2014.2363047.

[30] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Standards*, vol. 49, no. 6, pp. 409–436, Dec. 1952, doi: 10.6028/jres.049.044.

[31] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-94-125, Aug. 1994.

[32] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for Solution Linear Systems: Building Blocks for Iterative Methods*, 2nd ed., Philadelphia, PA, USA: SIAM, 1994, pp. 12–31.

[33] E. Carson and N. J. Higham, "A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems," *SIAM J. Sci. Comput.*, vol. 39, no. 6, pp. 2834–2856, Jan. 2017, doi: 10.1137/17m1122918.

[34] E. Carson and N. J. Higham, "Accelerating the solution of linear systems by iterative refinement in three precisions," *SIAM J. Sci. Comput.*, vol. 40, no. 2, pp. 817–847, Jan. 2018, doi: 10.1137/17m1140819.

[35] A. Haidar, H. Bayraktar, S. Tomov, J. Dongarra, and N. J. Higham, "Mixed-precision iterative refinement using tensor cores on GPUs to accelerate solution of linear systems," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 476, no. 2243, Nov. 2020, Art. no. 20200110, doi: 10.1098/rspa.2020.0110.

[36] A. Haidar, S. Tomov, J. Dongarra, and N. J. Higham, "Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage, Anal.*, Nov. 2018, pp. 603–613, doi: 10.1109/SC.2018.00036.

[37] H. Anzt, V. Heuveline, and B. Rocker, "Mixed precision iterative refinement methods for linear systems: Convergence analysis based on Krylov subspace methods," *Appl. Parallel Sci. Comput.*, vol. 7134, no. 2, pp. 237–247, 2012, doi: 10.1007/978-3-642-28145-7_24.

[38] S. Gratton, E. Simon, D. Titley-Peloquin, and P. Toint, "Exploiting variable precision in GMRES," 2019, *arXiv:1907.10550*.

[39] N. Lindquist, P. Luszczek, and J. Dongarra, "Improving the performance of the GMRES method using mixed-precision techniques," 2020, *arXiv:2011.01850*.

[40] A. Altinkaynak, "An efficient sparse matrix-vector multiplication on CUDA-enabled graphic processing units for finite element method simulations," *Int. J. Numer. Methods Eng.*, vol. 110, no. 1, pp. 57–78, Apr. 2017, doi: 10.1002/nme.5346.

[41] S. Filippone, V. Cardellini, D. Barbieri, and A. Fanfarillo, "Sparse matrix-vector multiplication on GPGPUs," *ACM Trans. Math. Softw.*, vol. 43, no. 4, pp. 1–49, Jan. 2017, doi: 10.1145/3017994.

[42] R. Helfenstein and J. Koko, "Parallel preconditioned conjugate gradient algorithm on GPU," *J. Comput. Appl. Math.*, vol. 236, no. 15, pp. 3584–3590, Sep. 2012, doi: 10.1016/j.cam.2011.04.025.

[43] A. F. P. de Camargos, V. C. Silva, J.-M. Guichon, and G. Munier, "Efficient parallel preconditioned conjugate gradient solver on GPU for FE modeling of electromagnetic fields in highly dissipative media," *IEEE Trans. Magn.*, vol. 50, no. 2, pp. 569–572, Feb. 2014, doi: 10.1109/TMAG.2013.2285091.

[44] R. Couturier and S. Domas, "Sparse systems solving on GPUs with GMRES," *J. Supercomput.*, vol. 59, no. 3, pp. 1504–1516, Mar. 2012, doi: 10.1007/s11227-011-0562-z.

[45] R. Li and Y. Saad, "GPU-accelerated preconditioned iterative linear solvers," *J. Supercomput.*, vol. 63, no. 2, pp. 443–466, Feb. 2013, doi: 10.1007/s11227-012-0825-3.

[46] K. Shibanuma and T. Utsunomiya, "Evaluation on reproduction of priori knowledge in XFEM," *Finite Elements Anal. Design*, vol. 47, no. 4, pp. 424–433, Apr. 2011, doi: 10.1016/j.finel.2010.11.007.

[47] M. J. Goovaerts, O. L. Gebizlioglu, I. Bayramoglu, and E. Akyildiz, "Preface," *J. Comput. Appl. Math.*, vol. 235, no. 16, pp. 4517–4518, Jun. 2011, doi: 10.1016/j.cam.2011.02.020.

[48] J. M. Jin and D. J. Riley, *Finite Element Analysis of Antennas and Arrays*. Hoboken, NJ, USA: Wiley-IEEE Press, 2009, pp. 55–60.

[49] H. Na and E. Lee, "Electromagnetic scattering analysis of 3D dielectric multi-coated structures using finite element method," *J. Korean Inst. Electromagn. Eng. Sci.*, vol. 34, no. 4, pp. 263–272, Apr. 2023, doi: 10.5515/kjkiees.2023.34.4.263.
[50] A. Chatterjee, J. M. Jin, and J. L. Volakis, "Edge-based finite elements and vector ABCs applied to 3-D scattering," *IEEE Trans. Antennas Propag.*, vol. 41, no. 2, pp. 221–226, Feb. 1993, doi: 10.1109/8.214614.
[51] Z. S. Sacks, D. M. Kingsland, R. Lee, and J.-F. Lee, "A perfectly matched anisotropic absorber for use as an absorbing boundary condition," *IEEE Trans. Antennas Propag.*, vol. 43, no. 12, pp. 1460–1463, Dec. 1995, doi: 10.1109/8.477075.
[52] R. Rumpf, *Electromagnetic and Photonic Simulation for the Beginner: Finite-Difference Frequency-Domain in MATLAB*. Norwood, MA, USA: Artech House, 2014, pp. 141–160.
[53] W. C. Chew and J. M. Jin, "Perfectly matched layers in the discretized space: An analysis and optimization," *Electromagnetics*, vol. 16, no. 4, pp. 325–340, Jul. 1996, doi: 10.1080/02726349608908483.
[54] F. Collino and P. B. Monk, "Optimizing the perfectly matched layer," *Comput. Methods Appl. Mech. Eng.*, vol. 164, nos. 1–2, pp. 157–171, Oct. 1998, doi: 10.1016/s0045-7825(98)00052-8.
[55] M. Movahhedi, A. Abdipour, H. Ceric, A. Sheikholeslami, and S. Selberherr, "Optimization of the perfectly matched layer for the finite-element time-domain method," *IEEE Microw. Wireless Compon. Lett.*, vol. 17, no. 1, pp. 10–12, Jan. 2007, doi: 10.1109/LMWC.2006.887240.
[56] A. Modave, E. Delhez, and C. Geuzaine, "Optimizing perfectly matched layers in discrete contexts," *Int. J. Numer. Methods Eng.*, vol. 99, no. 6, pp. 410–437, Aug. 2014, doi: 10.1002/nme.4690.
[57] O. Ozgun and M. Kuzuoglu, *MATLAB-Based Finite Element Programming in Electromagnetic Modeling*. Boca Raton, FL, USA: CRC Press, 2018, pp. 376–392.
[58] D. B. Kirk and W. H. Wen-Mei, *Programming Massively Parallel Processors: A Hands-on Approach*, 2nd ed., San Mateo, CA, USA: Morgan Kaufmann, 2013, pp. 41–62.
[59] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003, pp. 229–258.
[60] H. A. van der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 13, no. 2, pp. 631–644, Mar. 1992, doi: 10.1137/0913035.

MINCHEOL JO received the B.S. and M.S. degrees in electrical engineering from Incheon National University, in 2022 and 2024, respectively. His research interests include finite element method for electromagnetic analysis and computational electromagnetics.

WOOBIN PARK received the B.S. and M.S. degrees in electrical engineering from Incheon National University, in 2021 and 2023, respectively, where he is currently pursuing the Ph.D. degree. His research interests include finite element method for electromagnetic analysis and computational electromagnetics.

MOONSEONG KIM received the M.S. degree in mathematics and the Ph.D. degree in electrical and computer engineering from Sungkyunkwan University, South Korea, in August 2002 and February 2007, respectively. He was a Research Professor with Sungkyunkwan University, in 2007. From December 2007 to October 2009, he was a Research Associate with the Department of ECE and the Department of CSE, Michigan State University, USA. He was the Deputy Director and a Patent Examiner with Korean Intellectual Property Office, Daejeon, South Korea, from October 2009 to August 2018. In September 2018, he joined Seoul Theological University, Bucheon, South Korea, where he is currently an Associate Professor and the Head of the Department of IT Convergence Software. His research interests include mobile and sensor networks with intelligence and autonomicity, machine learning and artificial intelligence, and numerical analysis.

WOOCHAN LEE received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, South Korea, in 2002 and 2005, respectively, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2016. He was commissioned as a full-time Lecturer and a First Lieutenant with Korea Military Academy, Seoul, from 2005 to 2008. He was the Deputy Director and a Patent Examiner with Korean Intellectual Property Office, Daejeon, South Korea, from 2004 to 2017. In 2017, he joined the Department of Electrical Engineering, Incheon National University, Incheon, South Korea, where he is currently an Associate Professor. His current research interests include computational electromagnetics, numerical analysis, and the IoT applications with machine learning.