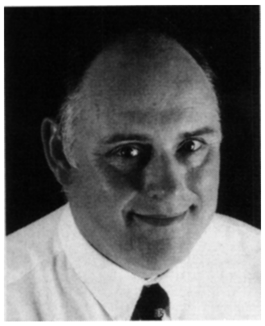

David B. Davidson
Dept. E&E Engineering
University of Stellenbosch
Stellenbosch 7600, South Africa
Tel: +27 21 808 4458
Fax: +27 21 808 4981
E-mail: davidson@sun.ac.za

Foreword by the Editor

In the December, 2007, column, the Finite-Difference Frequency-Domain Method was reviewed, and the issue of constructing an efficient pre-conditioner for the three-dimensional case was considered. This issue's first contribution continues the FDFD theme, looking at some methods for increasing the computational efficiency of the algorithm, with particular reference to memory usage and computation time.

There is also a second, shorter, contribution this month, on finding multiple roots of polynomials.

As always, we thank all the authors for their contributions.

An Algorithm for Efficient Solution of Finite-Difference Frequency-Domain (FDFD) Methods

Veysel Demir[1], Erdogan Alkan[2], Atef Z. Elsherbeni[3], and Ercument Arvas[4]

[1]Department of Electrical Engineering, Northern Illinois University
DeKalb, IL 60115 USA
Tel: +1 (815) 753-8039; Fax: +1 (815) 753-1289; E-mail: demir@ceet.niu.edu

[2]Department of Electrical Engineering and Computer Science, Syracuse University
Syracuse, NY 13244 USA
Tel: +1 (315) 431-7295; E-mail: ealkan@syr.edu

[3]Department of Electrical Engineering, University of Mississippi
University, MS 38677 USA
Tel: +1 (662) 915-5382; Fax: +1 (662) 915-7231; E-mail: atef@olemiss.edu

[4]Department of Electrical Engineering and Computer Science, Syracuse University
Syracuse, NY 13244 USA
Tel: +1 (315) 443-4430; Fax: +1 (315) 443-4936; E-mail: earvas@syr.edu

Abstract

Finite-Difference Frequency-Domain methods (FDFD) require solution of large linear systems of equations. These large systems are represented by matrix equations including highly sparse coefficient matrices, and they can often only be solved by using iterative methods. This paper presents an algorithm in which the matrix-equation solution approach in an iterative method is replaced by a multi-step solution process. Instead of using a coefficient matrix, the coefficients in the FDFD formulations are kept as three-dimensional arrays, and they are treated as operators. The algorithm is used together with the Bi-Conjugate Gradients Stabilized (*BICGSTAB*) method. This is applied to a three-dimensional FDFD method to solve for scattering from dielectric objects. It is also applied to two other FDFD methods (a single-grid and a double-grid FDFD) to solve for scattering from chiral objects. It has been shown that the presented algorithm effectively reduces the solution time and memory requirements.

Keywords: Numerical analysis; algorithms; chiral media; finite difference methods; iterative methods; FORTRAN; electromagnetic scattering

1. Introduction

Numerical solution of systems represented by partial differential equations (PDEs) often requires solution of large linear systems of equations. The Finite-Difference Frequency-Domain (FDFD) Method is a numerical-analysis technique based on the partial-derivative form of Maxwell's curl equations in the frequency domain. An FDFD formulation can be obtained by approximating the partial derivatives in the curl equations by finite differences on a staggered Yee grid [1]. Such FDFD formulations were used in [2] and [3] to solve for scattering from dielectric objects, and in [4] and [5] to solve for scattering from chiral objects. The resulting linear systems of equations are very large and highly sparse, since only the interactions between the neighboring field components are considered in the equations. The solution of such large systems becomes very costly in terms of computer time and memory, if direct linear-system equation solvers, such as the Gaussian elimination method, are used. These large systems can often only be solved by iterative methods.

There are several iterative techniques that have been proposed for solving linear systems [6, 7]. Among these techniques, the Generalized Minimal Residual (GMRES) Method [8] and the Bi-Conjugate Gradients Stabilized (*BICGSTAB*) [9] method are the most commonly used techniques for numerical solution of Maxwell's equations. The convergence rates of these iterative techniques are generally very slow, and some techniques are applied to speed up the convergence rates. The most common technique to improve the convergence rate is preconditioning the sparse linear system. Preconditioners can be derived from knowledge of the original physical problems from which the linear system arises [6]. Although several types of preconditioning techniques are available in the literature [7], determining an efficient preconditioner for a given system is a complicated topic, and usually requires extensive research. It should also be noted that using a preconditioner in an iterative method incurs some extra cost, both initially, for the setup, and per iteration, for applying it. There is a tradeoff between the cost of constructing and applying the preconditioner, and the gain in convergence speed [7]. The introduction of new preconditioners has been the subject of several studies dealing with numerical solution of Maxwell's equations.

A linear system can be expressed in the form of a matrix equation,

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

where A is the matrix of coefficients, x is the vector of unknowns, and b is the right-hand-side vector related to excitations. If needed to be described in simple terms, an iterative solver starts with an initial guess, x_0, and tries to minimize the residual,

$$r_k = b - Ax_k = b - b',$$

as the iterations proceed, where x_k is the solution at the kth iteration. As the residual minimizes, the x_k converge to the solution, x. This process requires a multiplication of A by x_k to produce the next residual. From the perspective of a user of an advanced iterative solver such as *BICGSTAB*, the solver is like a black box, as illustrated in Figure 1. The user only needs to supply a function that will perform the multiplication of A by x_k, and return the result, b', to the iterative solver. The iterative algorithm then internally calculates the new solution vector. The iteration proceeds until a convergence criterion is met.

Although a linear system can be expressed by a matrix equation as in Equation (1), during the iterative procedure it is not

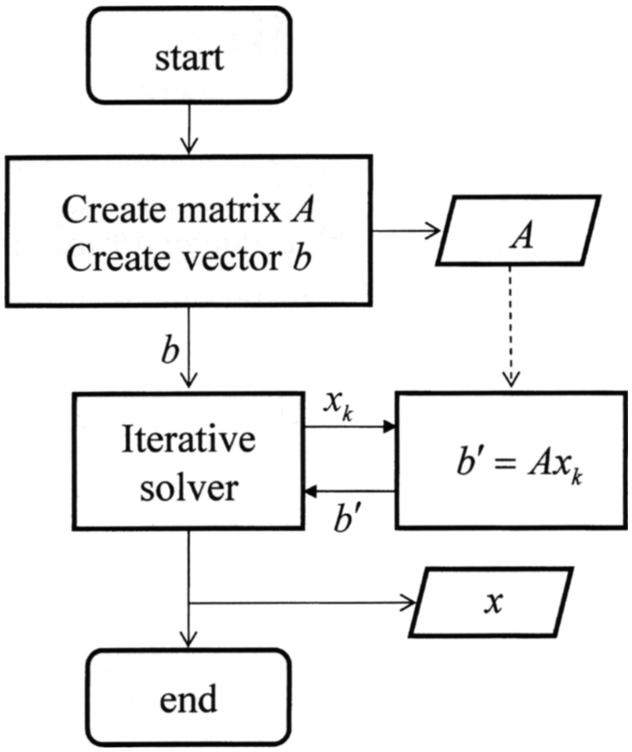

Figure 1. The procedure for calling an iterative solver.

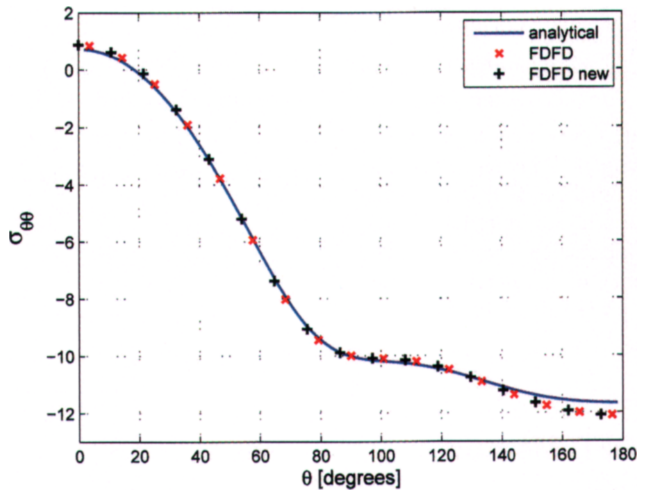

Figure 2. The bistatic radar cross section of a dielectric sphere.

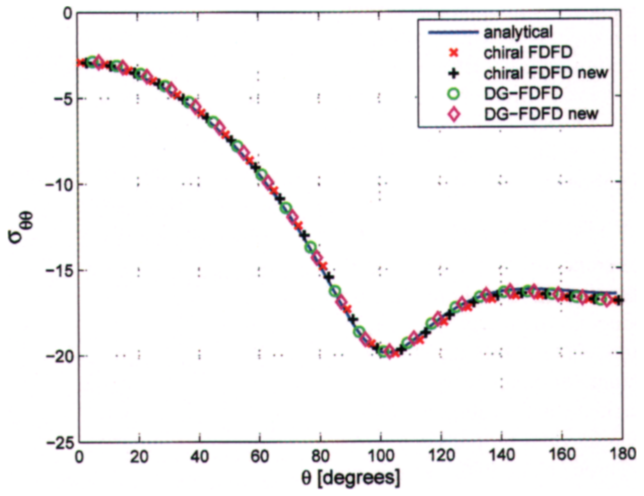

Figure 3. The co-polarized bistatic radar cross section of a chiral sphere.

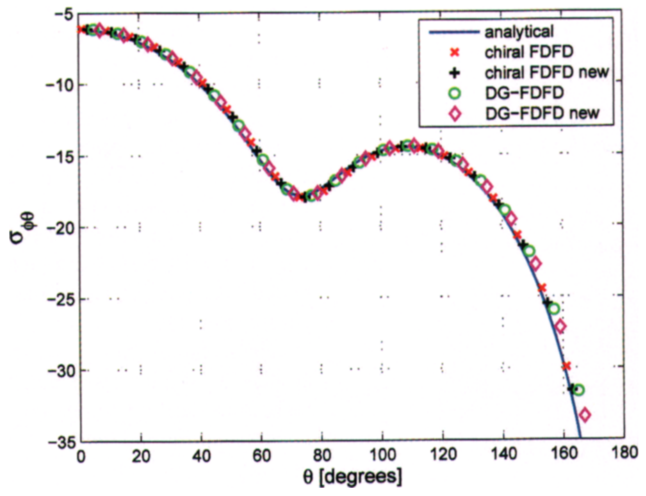

Figure 4. The cross-polarized bistatic radar cross section of a chiral sphere.

necessary to use the actual matrix, A, to store coefficients and to multiply it by x_k to calculate b'. It is only required to obtain b' for a given x_k. The aforementioned FDFD formulations [2-5] are examined in this context, and the Ax_k matrix-vector product is replaced by a multi-step algorithm. Due to the new algorithm, a significant reduction in computational time and memory usage has been achieved in the iterative solution of these FDFD methods. Furthermore, instead of one-dimensional arrays of coefficients, the use of three-dimensional arrays is introduced, which further improves the time and memory efficiency of the iterative solver. The details of this new approach are provided in the following sections.

2. Improvement of FDFD Method

2.1 FDFD Formulation

An FDFD method solves Maxwell's equations in the frequency domain. Such an FDFD method was developed and used in [2] and [3] to solve electromagnetic wave scattering problems from multiple three-dimensional conducting and dielectric objects. In this FDFD method, a three-dimensional problem space is divided into Yee cell's [1] using a Yee grid. The Yee grid is traditionally used in the Finite-Difference Time-Domain (FDTD) Method. On each Yee cell, six field components – namely E_x, E_y, E_z, H_x, H_y, and H_z – are located at discrete positions. This is done such that the electric-field components curl around the magnetic-field components, and the magnetic-field components curl around the electric-field components, naturally modeling Maxwell's curl equations. The derivation of the FDFD formulation starts with expressing the electric and magnetic fields in Maxwell's vector curl equations as the sum of incident and scattered fields. The incident field is the excitation field that propagates in a medium in which no scatterers exist. Three scalar PDEs are then obtained for the Cartesian coordinate system from each vector curl equation, resulting in a total of six equations. The partial derivatives in these six equations are expressed with finite differences by using central-difference approximations on a Yee grid. For instance, one of these equations has E_x as the pivot component, and it is written as

$$\begin{aligned}
&E_{scat,x}(i,j,k)-\frac{1}{j\omega\varepsilon_x(i,j,k)\Delta y}H_{scat,z}(i,j,k)\\
&+\frac{1}{j\omega\varepsilon_x(i,j,k)\Delta y}H_{scat,z}(i,j-1,k)\\
&+\frac{1}{j\omega\varepsilon_x(i,j,k)\Delta z}H_{scat,y}(i,j,k)\\
&-\frac{1}{j\omega\varepsilon_x(i,j,k)\Delta z}H_{scat,y}(i,j,k-1)\\
&=-\frac{\varepsilon_x(i,j,k)-\varepsilon_0}{\varepsilon_x(i,j,k)}E_{inc,x}(i,j,k),
\end{aligned}\tag{2}$$

where ω is the radian frequency for which a solution is sought, subscript *scat* denotes the scattered field, subscript *inc* denotes the incident field, index (i,j,k) indicates the cell in which the component is located, ε_x is the permittivity associated with E_x, and ε_0 is the free-space permittivity. Another equation hence has H_x as the pivot component, and it is written as

$$\begin{aligned}
&H_{scat,x}(i,j,k)+\frac{1}{j\omega\mu_x(i,j,k)\Delta y}E_{scat,z}(i,j+1,k)\\
&-\frac{1}{j\omega\mu_x(i,j,k)\Delta y}E_{scat,z}(i,j,k)\\
&-\frac{1}{j\omega\mu_x(i,j,k)\Delta z}E_{scat,y}(i,j,k+1)\\
&+\frac{1}{j\omega\mu_x(i,j,k)\Delta z}E_{scat,y}(i,j,k)\\
&=-\frac{\mu_x(i,j,k)-\mu_0}{\mu_x(i,j,k)}H_{inc,x}(i,j,k),
\end{aligned}\tag{3}$$

where μ_x is the permeability associated with H_x, and μ_0 is the free-space permeability. Similarly, the other four equations have pivot components of E_y, E_z, H_y, and H_z. In these equations, the scattered-field components are the unknowns.

One can use the magnetic-field pivot equations in the electric-field pivot equations, and reduce the number of equations from six to three. This eliminates the scattered magnetic-field components from the equations. The resulting equations were presented in [2]. Each of these equations has 13 terms on its left-hand side. Only the scattered electric-field components become the unknowns with this new set of equations. These equations are then combined to form a matrix equation in the form of Equation (1). If a three-dimensional problem space is composed of N cells, then the total number of unknowns becomes $3N$, and hence this is the size of the vectors x and b. The size of the coefficient matrix A is $(3N,3N)$. Actually, A is a highly sparse matrix, and it has 13 nonzero coefficients in its rows. The real size of A is thus $13\times 3N=39N$, which is the number of coefficients that need to be stored in the computer's memory. Special storage schemes are used to store such sparse matrices. One of these schemes is referred to as the *coordinate format*, and this is the scheme used in [2]. In this scheme the data structure consists of three arrays: (1) an array containing all the complex values of the nonzero elements of A; (2) an integer array containing their row indices; and (3) a second integer array containing their column indices [6]. All three arrays are of length $39N$. Although two other sparse storage schemes, called the compressed sparse row (CSR) format and the modified sparse row (MSR) format, are available and are slightly more efficient, the memory requirements are still very high for the storage of A. In this paper, the storage requirements for various techniques will be compared using the number of coefficients that need to be stored, without dealing with actual details, due to brevity.

2.2 Iterative Solver

The FDFD methods discussed in this paper were all programmed in the *FORTRAN* programming language. The iterative solver used was the "vanilla" version of *BICGSTAB* [10]. A *FORTRAN* implementation of the method can be obtained from the authors' Web site.

As discussed in the previous section, the user of the iterative solver – *BICGSTAB*, in this case – needs to supply a function, or a subroutine in *FORTRAN*, which will be called by the solver's program code, perform the operation $b' = Ax_k$, and return b' to the solver. If the coefficient matrix is stored using the coordinate format, such a function can be implemented as follows:

```
subroutine matvec(m, x, y)
use global, only: A, row, col, nnz
implicit none
integer m; complex*16 x(*), y(*);
integer i;
do i=1, nnz
    y(row(i)) = y(row(i)) + A(i)*x(col(i));
end do
end subroutine matvec
```

Here, the name of the subroutine is kept as `matvec` to be consistent with the implementation of the *BICGSTAB*. In this subroutine, `x` is the input corresponding to x_k, and `y` is the output corresponding to b'. The parameters `x` and `y` are one-dimensional arrays with a size of `m`. The parameters `A`, `row`, and `col` are the one-dimensional arrays used to store the value, row, and column information of the sparse A matrix, respectively. The parameter `nnz` is the number of nonzero coefficients in A, in other terms, the size of arrays `A`, `row`, and `col`.

BICGSTAB only needs to receive `y` from the `matvec` function. The method of storing the coefficients and calculating `y` is completely up to the user, in this case the programmer of the `matvec` function. In the given implementation, the coefficient arrays are stored in a global workspace. Due to the implied freedom in the method of calculation of `y` (i.e., b'), the FDFD formulation is revisited, and a more-efficient algorithm is developed.

2.3 The New Algorithm

Examining Equation (2), one can notice that the electric-field pivot equations can be cast in a matrix equation as

$$x_e + A_e x_h = b_e, \qquad (4)$$

where x_e is a vector for scattered electric-field components, x_h is a vector for scattered magnetic-field components, A_e is a coefficient matrix, and b_e is the right-hand-side vector. Similarly, as can be observed in Equation (3), the magnetic-field pivot equations can be cast in a matrix equation as

$$x_h + A_h x_e = b_h, \qquad (5)$$

where A_h is a coefficient matrix, and b_h is the right-hand-side vector. The size of vectors x_e, x_h, b_e, and b_h is $3N$. The number of nonzero coefficients in A_e and A_h is $4 \times 3N = 12N$, since there are only four nonzero coefficients per row. After getting x_h from Equation (5) and using it in Equation (4), one can obtain

$$x_e - A_e A_h x_e = b_e - A_e b_h. \qquad (6)$$

The new algorithm is based on this equation. The right-hand side of Equation (6) is calculated as $b = b_e - A_e b_h$ before the iterative solver is called. During the iterations, the operation at the left-hand side is performed at every iteration, for a solution x_{ek} at the kth iteration in multiple steps, as

$$x_t = A_h x_e,$$
$$x_t = A_e x_t, \qquad (7)$$
$$b' = x_e - x_t,$$

where x_t is a vector used to store intermediate results. In this algorithm, the coefficient matrices A_e and A_h are stored separately, and the total number of coefficients that need to be stored is reduced to $24N$, which is a significant reduction from $39N$. A second improvement in the algorithm reduces this number further, as detailed next.

Examining Equation (2), one can notice that although there are four coefficients in the equation, there are actually two coefficient pairs, in which the pairs are different only by their signs. Equation (2) can be rewritten as

$$\begin{aligned} & E_{scat,x}(i,j,k) \\ & \quad + C_{exhz}(i,j,k)\left[H_{scat,z}(i,j-1,k) - H_{scat,z}(i,j,k)\right] \\ & \quad + C_{exhy}(i,j,k)\left[H_{scat,y}(i,j,k) - H_{scat,y}(i,j,k-1)\right] \\ & = -\frac{\varepsilon_x(i,j,k) - \varepsilon_0}{\varepsilon_x(i,j,k)} E_{inc,x}(i,j,k), \end{aligned} \qquad (8)$$

where $C_{exhz}(i,j,k)$ and $C_{exhy}(i,j,k)$ are coefficients. This form of the equation implies that the number of coefficients can be reduced to two per equation, and the total number of coefficients that need to be stored is $12N$. These coefficients are indexed with (i,j,k) for a three-dimensional problem. It is therefore natural to store them as three-dimensional arrays in the computer's memory, instead of three one-dimensional arrays as is required by the coordinate format. The use of three-dimensional arrays provides the following main advantages:

- The number of coefficients is reduced by half: $12N$ coefficients instead of $24N$.

- The use of one-dimensional row and column arrays is eliminated. These arrays store integer data, while the coefficient arrays store complex data. If 16 bytes are used for a complex datum and four bytes are used for an integer datum, the amount of memory needed per coefficient will reduce to 16 bytes from 24 bytes. The total memory saving then becomes almost 80% for storing the coefficients, compared to the unmodified FDFD.

- The code listing for `matvec` shows that the matrix-vector product is performed using a `for` loop. It is well known that a *FORTRAN* program will provide superior performance if the program uses array operations rather than `for` loops. As will be illustrated next, the electric-field components are also stored in three-dimensional arrays, and `matvec` is implemented completely using array operations.

As mentioned before, the input and output arrays, x and y, of the `matvec` function are stored and used as one-dimensional arrays in the *BICGSTAB* code. The array x includes the solution for scattered-field components $E_{scat,x}$, $E_{scat,y}$, and $E_{scat,z}$, arranged as the vector x_{ek}. In order to use it in three-dimensional array operations, the x array should be converted to three-dimensional arrays corresponding to the $E_{scat,x}$, $E_{scat,y}$, and $E_{scat,z}$ field components in the `matvec` function. A one-dimensional to three-dimensional array transformation is therefore required. Actually, since the x array includes the data for three field-component types, each of which is a three-dimensional array, a one-dimensional to four-dimensional array transformation is performed in the modified `matvec` function for the new algorithm. If a problem space is composed of $Nx \times Ny \times Nz = N$ cells, where Nx, Ny, and Nz are the numbers of cells in the x, y, and z directions, respectively, then the x is cast into a four-dimensional array as `x(Nx, Ny, Nz, 3)` from a one-dimensional array `x(3*N)`. This transformation is performed by defining x as a four-dimensional array, as illustrated in the following modified `matvec` subroutine. This transformation operation does not bring any additional computational cost, since in either form of x, the data allocation in the physical memory is the same: only the way it is being interpreted by the compiler is different. By also representing the field-component arrays as three-dimensional arrays embedded in a four-dimensional array, the implementation of the `matvec` subroutine is modified, based on the algorithm in Equation (7). A section of the code is shown in the following listing.

```
subroutine matvec(n,x,y)
use global
implicit none
integer n;
complex*16 x(Nx,Ny,Nz,3), y(Nx,Ny,Nz,3);
tmpx(:,1:Ny-1,1:Nz-1) = &
    Chxez(:,1:Ny-1,1:Nz-1) &
        *(+x(:,2:Ny,1:Nz-1,3) &
        -x(:,1:Ny-1,1:Nz-1,3))&
    + Chxey(:,1:Ny-1,1:Nz-1) &
        *(-x(:,1:Ny-1,2:Nz,2) &
        +x(:,1:Ny-1,1:Nz-1,2));
...
...
y(1:Nx,2:Ny,2:Nz,1) = &
        x(1:Nx,2:Ny,2:Nz,1) &
    -(Cexhz(1:Nx,2:Ny,2:Nz) &
        *(-tmpz(1:Nx,2:Ny,2:Nz) &
        +tmpz(1:Nx,1:Ny-1,2:Nz)) &
    + Cexhy(1:Nx,2:Ny,2:Nz) &
        *(+tmpy(1:Nx,2:Ny,2:Nz) &
        -tmpy(1:Nx,2:Ny,1:Nz-1)));
...
...
end subroutine matvec
```

Here, tmpx, tmpy, and tmpz are three-dimensional arrays storing the intermediate results as implied by x_t in Equation (7).

2.4 Validation of the New Algorithm

In order to check the performance of the proposed algorithm in terms of computation time, a scattering problem was solved using the old and the new algorithms. A dielectric sphere with a dielectric constant of 4.0 was illuminated by an x-polarized plane wave at 1 GHz, traveling in the z direction. The radius of the sphere was 10 cm. The bistatic radar cross section due to the incident wave was calculated using both algorithms, and the results were compared to the analytical solution [11] of the same problem. Figure 2 shows that there was good agreement between the FDFD solutions and the analytical solution.

In the FDFD calculations, the problem space was composed of $(N_x = 100,\ N_y = 100,\ N_z = 100) = 10^6$ cells. The total number of the scattered electric-field components – thus, the unknowns – was 3×10^6. The number of coefficients in the old algorithm was 39×10^6, and in the new algorithm it was 12×10^6. The calculation time for the old algorithm was recorded as 61 minutes, while for the new algorithm it was 46 minutes. The FDFD calculation time was reduced by 25% using the new algorithm. It should be noted that the simulation times for problems with the same size may vary significantly between different problems, since the convergence rate of the iterative solver may depend on some other factors, as well.

The FDFD calculations referenced in this paper were all performed on a computer with the Microsoft *XP Professional x64* edition operating system and an Intel Xeon CPU E5405 at 2 GHz.

3. Improvement of the Chiral FDFD Method

The FDFD formulation discussed in the previous section was developed to calculate scattering from three-dimensional dielectric and conducting objects, [2] and [3]. In [4], the use of the FDFD method was extended to solve for scattering from chiral objects. The derivation of the chiral FDFD formulation is very similar to the FDFD formulation. The main difference is in the constitutive relations on which they are based: the FDFD is based on constitutive relations for dielectric media, while the chiral FDFD formulation is based on the constitutive relations for chiral media, which are given as

$$\begin{aligned} \overline{D} &= \varepsilon \overline{E} - j\kappa\sqrt{\varepsilon_0\mu_0}\,\overline{H}, \\ \overline{B} &= \mu \overline{H} + j\kappa\sqrt{\varepsilon_0\mu_0}\,\overline{E}, \end{aligned} \tag{9}$$

where κ is the chirality of the medium under consideration. The FDFD formulation is derived for a Yee grid on which the electric- and magnetic-field components are located at different discrete positions. However, as can be seen in Equation (9), the electric- and magnetic-field components are directly coupled to each other through the constitutive relations. This coupling requires the coexistence of electric- and magnetic-field components at the same spatial locations. To overcome this problem, electric-field components were averaged at the positions of the magnetic-field components, and magnetic-field components were averaged at the positions of the electric-field components, in [4]. Six equations, each of which is pivoted by E_x, E_y, E_z, H_x, H_y, or H_z, are then obtained for a cell. For instance, the equation for which E_x is a pivot reads

$$
\begin{aligned}
&E_{scat,x}(i,j,k)-\frac{1}{j\omega\varepsilon_x(i,j,k)\Delta y}H_{scat,z}(i,j,k)\\
&\quad+\frac{1}{j\omega\varepsilon_x(i,j,k)\Delta y}H_{scat,z}(i,j-1,k)\\
&\quad+\frac{1}{j\omega\varepsilon_x(i,j,k)\Delta z}H_{scat,y}(i,j,k)\\
&\quad-\frac{1}{j\omega\varepsilon_x(i,j,k)\Delta z}H_{scat,y}(i,j,k-1)\\
&\quad+\frac{\kappa_x(i,j,k)}{j\omega\varepsilon_x(i,j,k)c8}\times\big[H_{scat,x}(i,j,k)+H_{scat,x}(i,j,k-1)\\
&\qquad+H_{scat,x}(i,j,k)+H_{scat,x}(i+1,j,k-1)\\
&\qquad+H_{scat,x}(i,j-1,k)+H_{scat,x}(i,j-1,k-1)\\
&\qquad+H_{scat,x}(i+1,j-1,k)+H_{scat,x}(i+1,j-1,k-1)\big]\\
&=-\frac{\varepsilon_x(i,j,k)-\varepsilon_0}{\varepsilon_x(i,j,k)}E_{inc,x}(i,j,k)\\
&\quad-\frac{\kappa_x(i,j,k)}{j\omega\varepsilon_x(i,j,k)c8}\times\big[H_{inc,x}(i,j,k)+H_{inc,x}(i,j,k-1)\\
&\qquad+H_{inc,x}(i+1,j,k)+H_{inc,x}(i+1,j,k-1)\\
&\qquad+H_{inc,x}(i,j-1,k)+H_{inc,x}(i,j-1,k-1)\\
&\qquad+H_{inc,x}(i+1,j-1,k)+H_{inc,x}(i+1,j-1,k-1)\big],
\end{aligned}
\qquad (10)
$$

where c is the speed of light in free space. The other five equations read similarly. These equations are arranged to form a matrix equation as shown by Equation (1), where the vector x includes all of the scattered electric- and magnetic-field components. The size of the vectors x and b is $6N$. The sparse coefficient matrix A has 13 nonzero coefficients per row, so the total number of nonzero coefficients that need to be stored is $13\times 6N = 78N$. It is possible to use the magnetic-field pivot equations in the electric-field pivot equations and to obtain a matrix equation in which only the scattered electric-field components are unknowns, but this is not feasible, since the number of coefficients per row of A would become 52.

The new algorithm described in the previous section was applied to the chiral FDFD formulation derived in [4]. The procedure in Equation (7) was used to calculate b' for a given x_{ek} vector, which includes only the scattered electric-field components as unknowns. In [4], the coordinate format was used to store the coefficients of A, for which the number of coefficients was $78N$. Examining Equation (10), one can see that the number of coefficients that are different from each other is actually three. Using three-dimensional arrays to store the coefficients, the total number of coefficients is reduced to $3\times 6N = 18N$. With the elimination of the integer row and column arrays, the memory reduction becomes almost 85%.

In order to check the computational time reduction, a scattering problem was solved using the old and the new algorithms. The scattering problem was the same as the one presented in the previous section, except that the sphere had a nonzero chirality, $\kappa = 0.3$. The co- and cross-polarized bistatic radar cross sections due to the incident wave were calculated using both algorithms, and the results were compared to the analytical solution [11] of the same problem. Figure 3 shows the co-polarized radar cross sections, whereas Figure 4 shows the cross-polarized radar cross sections. Due to the optical-activity property of the chiral medium, the scattered field includes cross-polarization as well in the *x*-*z* plane. The results showed good agreement between the chiral FDFD solutions and the analytical solution.

In the chiral FDFD calculations, the problem space was composed of 10^6 cells. The total number of the scattered electric field components, and thus the unknowns, was 3×10^6. The number of coefficients in the old algorithm was 78×10^6, and in the new algorithm it was 18×10^6. The calculation time for the old algorithm was recorded as 307 minutes, while for the new algorithm it was 57 minutes. The FDFD calculation time was reduced by 80% using the new algorithm.

4. Improvement of the Double-Grid Chiral FDFD Method

As discussed in the previous section, the chiral constitutive relations require the coexistence of electric- and magnetic-field components, and this problem was overcome in [4] by averaging the fields on the Yee grid. As an alternative solution, a double-grid approach, referred to as the DG-FDFD, was presented in [5]. Instead of a single grid, a Yee grid and a transverse Yee grid are used, where like components of electric and magnetic fields from these grids coexist at the same spatial positions. The field components of these fields are coupled to each other. Due to the introduction of a second grid, the total number of field components has increased two-fold.

For the equation of the first grid in which the electric-field component, E_x, is the pivot, we have

$$
\begin{aligned}
&E_{scat,xa}(i,j,k)-\frac{1}{j\omega\varepsilon_{xa}(i,j,k)\Delta y}H_{scat,za}(i,j,k)\\
&\quad+\frac{1}{j\omega\varepsilon_{xa}(i,j,k)\Delta y}H_{scat,za}(i,j-1,k)\\
&\quad+\frac{1}{j\omega\varepsilon_{xa}(i,j,k)\Delta z}H_{scat,ya}(i,j,k)\\
&\quad-\frac{1}{j\omega\varepsilon_{xa}(i,j,k)\Delta z}H_{scat,ya}(i,j,k-1)\\
&\quad-\frac{j\kappa_{xa}}{\varepsilon_{xa}(i,j,k)}H_{scat,xb}(i,j,k)\\
&=-\frac{\varepsilon_{xa}(i,j,k)-\varepsilon_0}{\varepsilon_{xa}(i,j,k)}E_{inc,xa}(i,j,k)\\
&\quad\frac{j\kappa_{xa}}{\varepsilon_{xa}(i,j,k)}H_{inc,xb}(i,j,k),
\end{aligned}
\qquad (11)
$$

where the subscript a denotes the components on the first grid, and b denotes the components on the second grid. The other 11 equations read similarly for the field components in both grids [5]. These equations are combined to form a matrix equation as in Equation (1). The vectors x and b each have a size of $12N$, and the sparse matrix A has $6\times 12N = 72N$ nonzero coefficients.

The new algorithm was then applied to improve the efficiency of the DG-FDFD method, where the procedure in

Equation (7) was employed. The vector x_e includes the scattered electric-field components in both grids: the number of unknowns thus becomes $6N$. Since there are three distinct coefficients in Equation (11), the number of coefficients that needs to be stored reduces to $36N$, from $72N$. With the elimination of integer arrays of the coordinate format, the actual memory requirement for the coefficients is reduced by 66%.

In order to check the computational time reduction, the same scattering problem as the one presented in the previous section was used. The co- and cross-polarized bistatic radar cross sections due to the incident wave were calculated using the unmodified and modified DG-FDFD algorithms, and the results were compared to the analytical solution [11] of the same problem. Figure 3 shows the co-polarized radar cross sections, whereas Figure 4 shows the cross-polarized radar cross sections. The results showed good agreement among the DG-FDFD, chiral FDFD, and the analytical solutions.

In the DG-FDFD calculations, the problem space was composed of 10^6 cells. The total number of scattered electric-field components, and thus the unknowns, was 6×10^6. The number of coefficients using the coordinate format was 72×10^6, and using the three-dimensional arrays it was 36×10^6. The calculation time for the new algorithm in Equation (7) using the coordinate-format storage scheme was recorded as 200 minutes, while for the new algorithm using three-dimensional arrays it was 155 minutes. The FDFD calculation time was reduced by 22%, just by changing the data-storage scheme.

5. Conclusion

An algorithm to improve the time and memory efficiency of the iterative solution of FDFD methods has been presented. It has been shown that up to 80%, 85%, and 66% memory reductions can be achieved for the storage of coefficients arising in the FDFD, chiral FDFD, and DG-FDFD methods, respectively. The memory reductions achieved by the new algorithm are tabulated in Table 1. Meanwhile, very significant computational time reductions have also been observed with the test cases. The calculation times of the test cases are tabulated in Table 2. The use of efficient preconditioners could improve the simulation times even more. The prescribed algorithm also has the potential to improve the efficiency of other FDFD-like frequency-domain methods, such as the Finite-Element Method (FEM).

Table 1. The memory reduction achieved by using three-dimensional arrays instead of coordinate format for storing the coefficients.

Method	Memory Reduction
FDFD	80%
Chiral FDFD	85%
DG-FDFD	66%

Table 2. The recorded calculation times of the tests (minutes).

Method	Time
FDFD old algorithm	61
FDFD new algorithm	46
Chiral FDFD old algorithm	307
Chiral FDFD new algorithm	57
DG-FDFD new algorithm (coordinate format)	200
DG-FDFD new algorithm (three-dimensional arrays)	155

6. References

1. K. S. Yee, "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media," *IEEE Transactions on Antennas and Propagation*, **AP-14**, 3, May 1966, pp. 302-307.

2. M. H. Al Sharkawy, V. Demir, and A. Z. Elsherbeni, *Electromagnetic Scattering Using the Iterative Multiregion Technique, Synthesis Lectures on Computational Electromagnetics, First Edition*, San Rafael, CA, Morgan & Claypool Publishers, December 2007.

3. M. Al Sharkawy, V. Demir, and A. Z. Elsherbeni, "The Iterative Multi-Region Algorithm Using a Hybrid Finite Difference Frequency Domain and Method of Moments Techniques," *Progress in Electromagnetics Research (PIER)*, **57**, 2006, pp. 19-32.

4. L. Kuzu, V. Demir, A. Z. Elsherbeni, and E. Arvas, "Electromagnetic Scattering from Arbitrarily Shaped Chiral Objects Using the Finite Difference Frequency Domain Method," *Progress in Electromagnetics Research (PIER)*, **67**, 2007, pp. 1-24.

5. E. Alkan, V. Demir, A. Z. Elsherbeni, and E. Arvas, "Electromagnetic Scattering from Chiral Objects Using Double-Grid Finite-Difference Frequency-Domain (DG-FDFD) Method," 2009 IEEE MTT-S International Microwave Symposium *Digest*, 2009, pp. 173-176.

6. Y. Saad, *Iterative Methods for Sparse Linear Systems*, Philadelphia, PA, Society for Industrial and Applied Mathematics, 2003.

7. 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 the Solution of Linear Systems: Building Blocks for Iterative Methods, Second Edition*, Philadelphia, PA, Society for Industrial and Applied Mathematics, 1994.

8. Y. Saad and M. H. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, **7**, 3, July 1986, pp. 856-869.

9. H. A. van der Vorst, "BI-CGSTAB: A Fast and Smoothly Converging Variant of BI-CG for the Solution of Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, **13**, 2, March 1992, pp. 631-644.

10. G. L. G. Sleijpen and D. R. Fokkema, "BICGSTAB(l) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum," *Electronic Transactions on Numerical Analysis*, **1**, 1993, pp. 11-32.

11. V. Demir, A. Z. Elsherbeni, D. Worasawate, and Ercument Arvas, "A Graphical User/Interface (GUI) for Plane-wave Scattering from a Conducting, Dielectric, or Chiral Sphere," *IEEE Antennas and Propagation Magazine*, **46**, 5, October 2004, pp. 94-99.