Electromagnetic simulation of radiometer calibration targets using Finite Difference Time Domain (FDTD) method

Srikumar Sandeep
srikumarsandeep@yahoo.com

Submitted in partial fulfillment of the requirements of
ECEN 5264
Electromagnetic Absorption, Scattering, and Propagation

Department of Electrical and Computer Engineering
University of Colorado at Boulder

May 1, 2009

ABSTRACT

In this paper, I outline the work related to Finite Difference Time Domain (FDTD) method that was done in this semester. The ultimate goal of this work is to perform coupled electromagnetic – thermal analysis of radiometer calibration targets. This report describes the preliminary work that was performed towards this end. Two dimensional Finite Difference Time Domain (FDTD) method was implemented in software along with Total Field / Scattered Field (TF /SF) formulation. The code was numerically validated by estimating the Radar Cross Section (RCS) of right circular metallic cylinder and comparing it with analytical result. The report also contain details of HFSS simulations of electromagnetic wave absorbing pyramids.

1.0 INTRODUCTION

Radiometer calibration target, as shown in figure 1 is an array of pyramids, where each pyramid is made of composite material. The composite material consists of carbonyl iron particle inclusions embedded in a dielectric resin. It is envisaged that a coupled electromagnetic – thermal analysis of these targets will be carried out using three dimensional FDTD. This is possible by solving both the Maxwell’s equation and Heat Transfer Equation (HTE) in the computational domain.

In order to achieve this goal, a well tested and stable FDTD code should be developed. This semester’s work concentrated on 2D FDTD. 3D FDTD code was written in C++ as a part of the term paper for fall 2008 antennas and electromagnetic radiation course [2]. The 3D FDTD code consisted of 3D FDTD – UPML implementation. However it did not incorporate Total Field / Scattered Field ( TF /SF ) formulation. TF / SF is crucial for this project because of the need to generate plane waves of arbitrary propagation direction.
This semester works can be summarized as follows: literature survey of FDTD applications in remote sensing, learning TF / SF formulation in FDTD, implementation of TF / SF in 1D FDTD, implementation of TF / SF in 2D FDTD, numerical validation of 2D FDTD – TF / SF code and HFSS simulations of wave absorbing pyramids.

2.0 APPLICATION OF FDTD TO REMOTE SENSING PROBLEMS

The finite difference time domain method is a time domain numerical method that can be used to numerically solve partial differential equations. FDTD method is extensively used in Computational Electromagnetic Modeling (CEM) [3]. The two major advantages of FDTD over frequency domain CEM methods is that it can provide broadband response of a system with a single simulation using a Gaussian pulse excitation and being a time domain method, FDTD can handle complex nonlinear systems. In this section, the application of FDTD to remote sensing and other problems relevant to the course is outlined.

FDTD simulation of electromagnetic wave propagation in the earth – ionosphere waveguide at the ELF, VLF range is reported in [4]. Lightning-generated electromagnetic wave phenomena have been explored using 2D FDTD in [5]. Modeling of impulsive ELF propagation within the global Earth – ionosphere cavity can be potentially used as electromagnetic precursors of major earthquakes and development of novel means for the remote sensing of underground oil and ore deposits [3]. FDTD analysis of Ground Penetrating Radar (GPR) antennas and related GPR problems can be extensively found in FDTD literature [6, 7].

FDTD can be used for studying the electromagnetic field propagation in random dielectric medium. This includes numerical validation of dielectric mixing formulas and evaluation of effective dielectric properties of composite materials [8, 9]. Novel materials such as Left-Handed Materials (LHM), Double Negative Meta - materials (DNG) can be analyzed for their dispersion relations and other electromagnetic properties. Lately FDTD has been employed for performing simulations of light scattering by ice crystals, aerosols etc. In this way, scattering particles of any dimension can be studied. Otherwise, geometrical optics approximation has to be used to analyze only particles of dimensions much larger than the incident wavelength. FDTD has been used to calculate scattering phase function of arbitrary shaped scatterers [10, 11, 12]. FDTD can also be used to estimate 2D and 3D Radar Cross Section (RCS) of arbitrary shaped bodies [13]. Simple examples of this procedure for estimating RCS of right circular and square cylinders are illustrated in this report.

As mentioned earlier, coupled electromagnetic – thermal analysis using FDTD is the main objective of this research project. A few papers relating to this work can be found in literature. This includes the application of FDTD for the simulation of microwave drilling [14]. Microwave
drilling is a novel method for drilling hard non–conductive materials. Similar techniques have been applied to industrial microwave heating problems [15]. These papers perform coupled FDTD – HTE simulation by using the fact that dissipated electromagnetic energy (dielectric loss, joule loss, magnetic loss) result in temperature fluctuations in the material. A few papers detailing the simulation of ferrite electromagnetic absorbers using FDTD is listed in the references [16, 17].

![Iron–epoxy radiometer calibration target for use with Polarimetric Scanning Radiometer (PSR).](http://cet.colorado.edu/instruments/psr/)

2.0 TOTAL-FIELD / SCATTERED FIELD (TF / SF) FORMULATION

Total field / Scattered field formulation is an incident source condition used in FDTD to generate uniform plane waves of arbitrary polarization, time dependence and wave vector [3, 18]. The TF / SF formulation is based on the linearity of the Maxwell’s equation. The total electric / magnetic field in the computational domain can be decomposed into the incident and the scattered components, as given by (1).

$$
\vec{E}_{tot} = \vec{E}_{inc} + \vec{E}_{scat} \quad ; \quad \vec{H}_{tot} = \vec{H}_{inc} + \vec{H}_{scat}
$$

The 2D FDTD computational space is shown in figure 2. The computational space consists of the problem space surrounded on four sides by an absorbing boundary called Uniaxial Perfectly Matched Layer (UPML). UPML is a hypothetical anisotropic medium capable of absorbing electromagnetic waves incident on it with minimal reflection [3]. The incident wave can be of arbitrary incident angle, frequency and polarization. Detailed derivation and implementation details of FDTD – UPML is three dimensions can be found in [3]. Two dimensional and three dimensional FDTD – UPML formulation and software implementation can be found in [19] and [2] respectively. Both of these reports are attached along with this paper for quick reference. This report does not attempt to repeat these formulation details to ensure brevity.
In figure 2, it can be seen that the UPML is surrounded by a Perfect Electric Conductor (PEC) layer. The internal problem space is further divided into two regions, namely the Total Field (TF) region and the Scattered Field (SF) region. This is the difference between a normal FDTD computational space and the FDTD – TF / SF computational space. The scatterers of arbitrary shape and constitutive parameters are located in the TF region. The TF and SF regions are separated by a nonphysical virtual rectangular boundary called TF / SF boundary. In TF / SF formulation, the field components stored in memory for the TF region is the total field and those for the SF region is the scattered field. The FDTD update equations will be same for both the regions. The only difference is that they will be operating on two different set of fields.

The only discrepancy arising from this approach is when FDTD update equations are applied at the TF / SF boundary. At the TF / SF boundary, the field on one side of the boundary is total field and on the other side is scattered field. Therefore, when the difference between these field components are taken to update the field quantity on the boundary, there exists an inconsistency. Therefore update equations at these TF / SF boundary points use the value of incident field at these points at the current time to remove this inconsistency. Incident wave is the wave that would be present if there were no scatterers in the TF region.
3.0 ONE DIMENSIONAL TF / SF

In this section, one dimensional TF / SF formulation is briefly described [3, 19].

Consider a uniform plane wave propagating in the +y direction, with the field components $E_z$ and $H_x$. Figure 3 shows the one dimensional FDTD grid representing this wave. The FDTD update equations can be derived by applying central difference approximations to Maxwell’s curl equations with respect to the discretized spatial and temporal coordinates [3, 19, 20]. The update equations for $E_z$ and $H_x$ are given by (2) and (3).

$$
E_z^{n+1} = E_z^n - \frac{\Delta t}{\varepsilon \Delta y} \left[ H_x^n |_{i+0.5} - H_x^n |_{i-0.5} \right]
$$

$$
H_x^{n+0.5} = H_x^n |_{i+0.5} - \frac{\Delta t}{\mu \Delta y} \left[ E_z^n |_{i+1} - E_z^n |_{i} \right]
$$

In the equations (2) and (3), $n, i$ represent the temporal and spatial discretization indices respectively. These update equations can be applied to the entire computational domain. The only exceptions are for $E_z^n |_{i_L+1}, E_z^n |_{i_R+1}, H_x^n |_{i_L-0.5}$ and $H_x^n |_{i_R+0.5}$. For these field components (i.e. electric field components on the TF / SF boundary and magnetic field components just outside the TF / SF boundary), we need to apply consistency conditions to ensure that we use either total or scattered field when we take the differences. To exemplify this, consider the update equation (2) applied to $E_z^n |_{i_L+1}$. It can be noticed in (2), that we will subtracting $H_x^{scat} |_{i_L-0.5}$ from
$H_x^{tot}|_{l_L}^{n+0.5}$. This should be avoided by using $H_x^{tot}|_{l_L}^{n+0.5} = H_x^{scat}|_{l_L}^{n+0.5} + H_x^{inc}|_{l_L}^{n+0.5}$ instead of $H_x^{scat}|_{l_L}^{n+0.5}$ in (2).

$$E_z|_{l_L}^{n+1} = E_z|_{l_L}^{n} - \frac{\Delta t}{\varepsilon \Delta y} [H_x|_{l_L}^{n+0.5} - H_x|_{l_L}^{n+0.5} - H_x^{inc}|_{l_L}^{n+0.5}]$$ (4)

In (4), it should be noted that $H_x|_{l_L}^{n+0.5}$ is a scattered field component. This can be seen in figure 3. The value stored in computer memory for this location is a scattered field value. Equation (4) can be expressed in a more compact form as follows.

$$E_z|_{l_L}^{n+1} = \{E_z|_{l_L}^{n+1}\} + \frac{\Delta t}{\varepsilon \Delta y} H_x^{inc}|_{l_L}^{n+0.5}$$ (5)

In (5), $\{E_z|_{l_L}^{n+1}\}$ refers to equation (2), i.e. the ordinary FDTD update equation. In a similar way the TF / SF consistency equations can be obtained for other exceptions. They are given by equations (6) – (8).

$$E_z|_{l_R}^{n+1} = \{E_z|_{l_R}^{n+1}\} - \frac{\Delta t}{\varepsilon \Delta y} H_x^{inc}|_{l_R}^{n+0.5}$$ (6)

$$H_x|_{l_L}^{n+0.5} = \{H_x|_{l_L}^{n+0.5}\} + \frac{\Delta t}{\mu \Delta y} E_z^{inc}|_{l_L}^{n}$$ (7)

$$H_x|_{l_R}^{n+0.5} = \{H_x|_{l_R}^{n+0.5}\} - \frac{\Delta t}{\mu \Delta y} E_z^{inc}|_{l_R}^{n}$$ (8)

The equations (2),(3) in the TF and SF regions and equations (5) – (8) at the TF / SF boundary constitutes the TF / SF formulation for 1D FDTD.

TF / SF was implemented in the 1D FDTD framework. The MATLAB® code is given in appendix A. In figure 4, simulation results of a Gaussian pulse incident on a PEC plane is shown. At 167 ps, a Gaussian pulse enters the TF region. It can be assumed that the pulse was travelling from the left SF region. Since the incident field is not stored in the SF region, the pulse cannot be observed in the left SF region. At 750 ps, the pulse hits the PEC plane and gets reflected. It can be seen that there is a pulse propagating in right SF region after the incident pulse strikes the PEC plane. This is the scattered pulse which when added to the incident pulse, makes the total field zero on the right side of the PEC plane. It should be noted that this phenomena is explained by the Ewald-Oseen extinction theorem [21]. The computational domain is terminated on either ends by an analytical absorbing boundary condition. 1D TF / SF do not have any practical importance, but the implementation of it is helpful in understanding more complex 2D and 3D TF / SF.
4.0 TWO DIMENSIONAL TF / SF

The 2D TF / SF is more complex than 1D TF / SF in the sense that the incident wave can have arbitrary propagation direction. This can be seen in figure 2, where the incident wave is denoted by the wave vector $\hat{k}$. Depending on $\hat{k}$, the incident wave can strike the TF / SF boundary at one of its four vertices. 2D FDTD can be formulated either in $TM_z$ or $TE_z$ propagation modes [3]. The derivation of the governing differential equations for these two modes is given in appendix B. The 2D FDTD computational space cannot be terminated by a simple analytical absorbing boundary condition as that was used for 1D FDTD. This is due to the fact that the incident angle of the wave striking the boundary can be varying. To eliminate this problem, the UPML as shown in figure 2 is used. FDTD – UPML formulation for two dimensional case can be found in [19].

In order to implement TF / SF for the $TM_z$ case, we need to know the incident wave component values at the TF / SF boundary and at grid points just outside the TF / SF boundary.
In figure 5, it can be seen that we need to know the incident field values of $E_z$ on the TF / SF boundary and that of $H_x$ and $H_y$ at the locations shown by blue and green arrows respectively. This is done by using an auxiliary incident wave 1D FDTD. The direction of this 1D grid depends on the wave vector $\hat{k}$. Suppose we want to estimate the incident wave electric field intensity at point $P$. Point $O$ can be one of the vertices of TF / SF boundary depending on the wave vector direction. Since we know the location of $O$, and the direction $\hat{k}$, we can find the scalar projection of $\overrightarrow{OP}$ on the direction $\hat{k}$. Using this distance, the location of 1D FDTD grid points $A1$ and $A2$ are found out. The electric field intensity at point $P$ can be then estimated by linear interpolation of the field values at points $A1$ and $A2$. Another very important aspect in 2D or 3D TF / SF implementation is numerical phase velocity anisotropy [3]. This topic is briefly described in appendix C.

The consistency conditions for 2D FDTD TF / SF can be formulated in a similar way as 1D TF / SF consistency equations were derived in section 3.0. These equations are listed in appendix D. This equation along with ordinary 2D FDTD – UPML update equations given by (21) – (26) in
[UGthesis] completes the TF / SF formulation in a 2D FDTD – UPML framework. The MATLAB® 2D FDTD – UPML – TF /SF code is given in appendix E.

5.0 NUMERICAL VALIDATION OF 2D TF / SF : 2D RCS

In order to numerically validate 2D TF / SF code, 2D RCS of metallic right circular cylinders of different electrical radii were estimated. The scattering pattern was then compared with analytical results. The cylinder’s circular cross-section was located in the center of the FDTD problem space. A $TM_z$ wave with sinusoidal time variation was excited at the left TF / SF boundary. The direction of propagation of the wave is along the $x – axis \ (i.e. \ \varphi = 0^\circ)$ . The wave strikes the cylinder and the scattered wave alone can be observed in the SF region. The simulation is allowed to run, until the transients are negligible and a steady state solution is reached. The steady state magnitude of electric field intensities on a circle of constant radius with its center on the center of the problem space is extracted. The square of this quantity will be proportional to power density along the corresponding direction. The normalized power density versus azimuthal angle is plotted on a polar plot.

Analytical expression for the scattered wave by a metallic right circular cylinder can be found in [20, 22]. The ratio of the scattered field to the incident field for a conducting cylinder is given by (9).

\[
\left| \frac{E_z^{\text{scat}}}{E_z^{\text{inc}}} \right|^2 = \sqrt{\frac{2}{\pi k \rho}} \sum_{n=-\infty}^{\infty} \frac{J_n(ka)}{H_n^{(2)}(ka)} e^{in\varphi}
\]  

From figure 6, it can be seen that, there is reasonable match between the analytical and FDTD simulated results for three different cylinder electrical radii. The slight discrepancy between the two results, which is common in any numerical method, can be attributed to round – off errors, discretization errors etc. It should be noted that we represent the circular cross section of the cylinder in the FDTD lattice by using square cells. This is called staircase approximation and can have considerable impact on the solution. This can be rectified by finer meshing. In figure 6, it can be observed that the scattered field is strong in the shadow region. This can be explained by using the Ewald-Oseen extinction theorem [21]. Moreover, the scattering pattern is more directional for electrically large cylinders. Figure 7 shows the electric field intensity plot in the case of a Gaussian pulse striking a square conducting cylinder.

6.0 HFSS ANALYSIS OF PYRAMIDAL ABSORBERS

HFSS (High Frequency Structural Simulator) is a Finite Element Method (FEM) based software used for CEM. HFSS was used to analyze the wave reflection properties of pyramidal absorbers. The absorbers were made of Emerson – Cuming Inc., CR – 114 radar absorbent material. This
material is characterized by relative permittivity $\varepsilon_r = 9 - j0.4$ and relative permeability $\mu_r = 1 - j0.5$. By virtue of Poynting’s theorem, the imaginary part of relative permittivity and permeability results in dielectric and magnetic loss respectively.

In HFSS 3D modeler, pyramids are constructed by sweeping the base square along the direction of the pyramid’s height. The draft angle for the sweep operation can be specified. This angle determines the height of the pyramid. Structural transformation operation of mirror / duplicate can be used to generate an array of pyramids from a single pyramid. This is followed by the addition of substrate layer beneath the pyramids. All the dimensions and angles are parameterized using HFSS project variables. Therefore, dimensions can be changed without having to manually draw the pyramids again. Figure 8, shows the 3D model that was constructed in HFSS. It can be seen that the structure is surrounded by periodic boundary condition (known as master / slave boundary condition in HFSS) on the four sides. The incident plane is generated on the top plane, shown by the red rectangle and the reflected field magnitude and hence the reflectivity can be calculated on the green rectangle.

![Figure 8: 3D model constructed in HFSS](image)

**Figure 6.** RCS of right circular cylinder. $ka = \frac{2\pi a}{\lambda}$
Figure 7. Electric field intensity plot of Gaussian pulse striking a square cylinder

Figure 8. HFSS 3D model for pyramidal absorber simulation
7.0 FUTURE WORK

The 2D FDTD code developed in this semester can handle lossy dielectric materials, i.e. materials with purely real values for permittivity, permeability and conductivity. In order to take care of the imaginary part of the permittivity and permeability the FDTD formulation need to changed [3,20]. This will enable the simulation of common materials such as water and also CR - 114. This will be followed by the implementation of Periodic Boundary Condition (PBC) in 2D FDTD. Periodic boundary condition will allow the simulation of an infinite pyramidal absorber array. Reflectivity of a 2D absorber layer can be numerically estimated for varying frequencies, pyramid depth, pyramid spacing and substrate depth. The results obtained can be compared with published results [1] and with HFSS results. If the results match, then the project can progress into 3D FDTD. Furthermore HTE will be implemented in 2D FDTD. In parallel to FDTD work, dielectric characterization of the CR – 114 composite material will be carried out using microwave measurements [23]. The transmission / reflection method (T/R) can be used to estimate the permittivity and permeability of the material. In this technique, a sample of the Material Under Test (MUT) is placed inside a hollow waveguide or a coaxial cable. By comparing the analytical expressions for scattering parameters to the experimentally obtained results, the constitutive parameters of the material can be estimated.

REFERENCES


APPENDIX A : 1D FDTD code with TF / SF formulation

% One dimensional FDTD with TF / SF
% S. Sandeep, 2009

L = 0.5;       % Length of the computational domain
N = 500;       % Number of segments along the y - axis
dy = L / N;
c = 3e8;
dt = dy / (2 * c);
muo = 4 * pi * 1e-7;
epso = 8.85e-12;
eta = 120 * pi;
T = 1200;      % Time duration

% Field components are Ez and Hx
% Number of Ez components = N + 1
% Number of Hy components = N
Ez = zeros(T, N + 1);
Hx = zeros(T, N);

%Hx(i1, N) is to the right of Ez(i1, N)

%iL - Left interface of the TF/SF boundary
%iR - Right interface of TF/SF boundary
% To the left of iL / right og iR - the fields stored are scattered field
% Else - total fields are stored
iL = round(N / 3);
iR = round(2 * N / 3);
iM = round(N / 2);
iPEC = round((iM + iR)/2);

%n = 0 --> E = 0
%n = 0.5 --> H = 0
for n = 2 : 0.5 : T
    if(ceil(n) ~= n)
        %n = 0.5, 1.5...Update Hx
        nH = floor(n) + 1;
        for j = 1 : N
            dEz = Ez(nH, j + 1) - Ez(nH, j);
            Hx(nH, j) = Hx(nH - 1, j) - (dt / (muo * dy)) * dEz;
            if(j == iL)
                Ezinc = exp(-((((n - 0.5)*dt - 100*dt)/(40*dt))^2 ));
                Hx(nH, j) = Hx(nH, j) + (dt / (muo * dy)) * Ezinc;
            end
            if(j == iR)
                t = n - 0.5 - 2 * (iR - iL);
                Ezinc = exp(-(((t * dt - 100 * dt) / (40*dt))^2 ));
                Hx(nH, j) = Hx(nH, j) - (dt / (muo * dy)) * Ezinc;
            end
        end
    else
        nE = n + 1;
        % Update Ez
    end
end
for i = 1 : N + 1
  %ABC on both the left and right ends
  if(i == 1)
    Ez(nE,i) = Ez(nE - 2, i + 1);
  elseif(i == N + 1)
    Ez(nE,i) = Ez(nE - 2, i - 1);
  else
    dHx = Hx(nE - 1, i) - Hx(nE - 1, i - 1);
    %Put a PEC slab
    if(i == iPEC)
      Ez(nE,i) = 0;
    else
      Ez(nE,i) = Ez(nE - 1, i) - (dt / (eps0 * dy)) * dHx;
    end
  end
  if(i == iL)
    Hxinc = exp(-((n - 1.5)*dt - 100*dt)/(40*dt)^2 ));
    Ez(nE,i) = Ez(nE, i) + (dt / (eps0 * dy)) * (Hxinc / eta);
  end
  if(i == iR)
    t = n - 0.5 - 2*(iR - iL) - 1;
    Hxinc = exp(-(((t * dt - 100 * dt) / (40*dt))^2 ));
    Ez(nE,i) = Ez(nE, i) - (dt / (eps0 * dy)) * (Hxinc / eta);
  end
end
end
end
APPENDIX B : Governing equations for 2D FDTD propagation modes

Maxwell’s curl equation in time domain is decomposed into 6 scalar equations. Maxwell’s equation in linear, isotropic, nondispersive, lossy material.

\[ \nabla \times \vec{E} = -\mu \frac{\partial \vec{H}}{\partial t} - (\vec{M}_{\text{source}} + \sigma^* \vec{H}) \]
\[ \nabla \times \vec{H} = \varepsilon \frac{\partial \vec{E}}{\partial t} + (\vec{J}_{\text{source}} + \sigma \vec{E}) \]

System of six coupled scalar equations

\[ \frac{\partial H_x}{\partial t} = \frac{1}{\mu} \left[ \frac{\partial E_y}{\partial y} - \frac{\partial E_z}{\partial z} - (M_{\text{source}_x} + \sigma^* H_x) \right] \]
\[ \frac{\partial H_y}{\partial t} = \frac{1}{\mu} \left[ \frac{\partial E_z}{\partial z} - \frac{\partial E_x}{\partial x} - (M_{\text{source}_y} + \sigma^* H_y) \right] \]
\[ \frac{\partial H_z}{\partial t} = \frac{1}{\mu} \left[ \frac{\partial E_x}{\partial x} - \frac{\partial E_y}{\partial y} - (M_{\text{source}_z} + \sigma^* H_z) \right] \]
\[ \frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon} \left[ \frac{\partial H_y}{\partial y} - \frac{\partial H_z}{\partial z} - (J_{\text{source}_x} + \sigma E_x) \right] \]
\[ \frac{\partial E_y}{\partial t} = \frac{1}{\varepsilon} \left[ \frac{\partial H_z}{\partial z} - \frac{\partial H_x}{\partial x} - (J_{\text{source}_y} + \sigma E_y) \right] \]
\[ \frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon} \left[ \frac{\partial H_x}{\partial x} - \frac{\partial H_y}{\partial y} - (J_{\text{source}_z} + \sigma E_z) \right] \]

Partial derivatives with respect to z-axis is zero.

\[ \frac{\partial H_x}{\partial t} = \frac{1}{\mu} \left[ \frac{\partial E_y}{\partial y} - (M_{\text{source}_x} + \sigma^* H_x) \right] \]
\[ \frac{\partial H_y}{\partial t} = \frac{1}{\mu} \left[ \frac{\partial E_z}{\partial z} - (M_{\text{source}_y} + \sigma^* H_y) \right] \]
\[ \frac{\partial H_z}{\partial t} = \frac{1}{\mu} \left[ \frac{\partial E_x}{\partial x} - (M_{\text{source}_z} + \sigma^* H_z) \right] \]
\[ \frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon} \left[ \frac{\partial H_y}{\partial y} - (J_{\text{source}_x} + \sigma E_x) \right] \]
\[ \frac{\partial E_y}{\partial t} = \frac{1}{\varepsilon} \left[ \frac{\partial H_z}{\partial z} - (J_{\text{source}_y} + \sigma E_y) \right] \]
\[ \frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon} \left[ \frac{\partial H_x}{\partial x} - (J_{\text{source}_z} + \sigma E_z) \right] \]

2 propagation modes $TE_z, TM_z$

$TE_z : E_x, E_y$ and $H_z$
\[
\begin{align*}
\frac{\partial H_z}{\partial t} &= \frac{1}{\mu} \left[ \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} - (M_{\text{source} \ z} + \sigma^* H_z) \right] \\
\frac{\partial E_x}{\partial t} &= \frac{1}{\varepsilon} \left[ \frac{\partial H_z}{\partial y} - (J_{\text{source} \ x} + \sigma E_x) \right] \\
\frac{\partial E_y}{\partial t} &= \frac{1}{\varepsilon} \left[ -\frac{\partial H_z}{\partial x} - (J_{\text{source} \ y} + \sigma E_y) \right]
\end{align*}
\]

**TM_z : H_x, H_y and E_z**
\[
\begin{align*}
\frac{\partial H_x}{\partial t} &= \frac{1}{\mu} \left[ -\frac{\partial E_z}{\partial y} - (M_{\text{source} \ x} + \sigma^* H_x) \right] \\
\frac{\partial H_y}{\partial t} &= \frac{1}{\mu} \left[ \frac{\partial E_z}{\partial x} - (M_{\text{source} \ y} + \sigma^* H_y) \right] \\
\frac{\partial E_z}{\partial t} &= \frac{1}{\varepsilon} \left[ \frac{\partial H_z}{\partial x} - \frac{\partial H_x}{\partial y} - (J_{\text{source} \ z} + \sigma E_z) \right]
\end{align*}
\]
APPENDIX C: Numerical phase velocity anisotropy

Numerical dispersion is a phenomenon by which the FDTD algorithms for Maxwell’s curl equations cause nonphysical dispersion of the simulated waves in a free-space computational lattice. The variation of the phase velocity with respect to frequency is a function of propagation direction and grid discretization. The numerical phase velocity for a constant frequency wave can be different for different propagation directions. This phenomenon is called numerical phase velocity anisotropy.

The numerical dispersion relation is derived by substituting a monochromatic, plane-wave solution into the system of finite difference equations. This results in an algebraic equation that relates the numerical wave vector to wave frequency, time and spatial discretization. Using this numerical dispersion, a method for estimating the anisotropic numerical phase velocity can be formulated. The iterative procedure is given below.

\[ \tilde{k}_{i+1}(\phi) = k_i(\phi) - \frac{sin^2(Ak_i) + sin^2(Bk_i) - C}{Asin(2Ak_i) + Bsin(2Bk_i)} \]

\[ A = \frac{\Delta cos\phi}{2}, \quad B = \frac{\Delta sin\phi}{2}, \quad C = \frac{1}{S^2}sin^2\left(\frac{\pi S}{N_\lambda}\right) \]

where \( \tilde{k} \) is the numerical wave vector and \( \phi \) is the angle of wave propagation. \( S = \frac{c\Delta t}{\Delta} \) is called the courant stability factor. \( \Delta \) and \( N_\lambda = \frac{\lambda_\circ}{\Delta} \) are the spatial discretional interval and grid sampling density respectively. Numerical phase velocity is given by \( \bar{v}_p(\phi) = \frac{\omega}{k(\phi)} \).
APPENDIX D : 2D FDTD TF / SF consistency conditions

1. $H_y, B_y$ consistency conditions

\[ B_y^{n+1}_{i,j,0.5} = C_{1By} B_y^{n+1}_{i,j,0.5} + C_{2By} [E_z^{n+1}_{i+0.5,j,0.5} - E_z^{n+1}_{i-0.5,j+0.5}] \]

\[ C_{1By} = \frac{2 \varepsilon \kappa_z - \sigma_z \Delta t}{2 \varepsilon \kappa_z + \sigma_z \Delta t} \]

\[ C_{2By} = \frac{2 \varepsilon \Delta t}{2 \varepsilon \kappa_z \Delta x + \sigma_z \Delta t \Delta x} \]

$B_y(i_0 - 0.5, j_0 \leq j \leq j_1)$

\[ B_y^{n+1}_{i,j,0.5+0.5} = C_{1By} B_y^{n+1}_{i,j,0.5} + C_{2By} [E_z^{n+1}_{i,j,0.5+0.5} - E_z^{n+1}_{i,j,0.5-0.5}] - C_{2By} E_{z,inc}^{n+1}_{i,j,0.5+0.5} \]

$B_y(i_1 + 0.5, j_0 \leq j \leq j_1)$

\[ B_y^{n+1}_{i,j,0.5+0.5} = C_{1By} B_y^{n+1}_{i,j,0.5} + C_{2By} [E_z^{n+1}_{i+0.5,j,0.5} - E_z^{n+1}_{i-0.5,j+0.5}] + C_{2By} E_{z,inc}^{n+1}_{i,j,0.5+0.5} \]

2. $H_x, B_x$ consistency conditions

\[ B_x^{n+1}_{i+0.5,j} = C_{1Bx} B_x^{n+1}_{i+0.5,j} - C_{2Bx} [E_z^{n+1}_{i+0.5,j,0.5} - E_z^{n+1}_{i-0.5,j-0.5}] \]

\[ C_{1Bx} = \frac{2 \varepsilon \kappa_y - \sigma_y \Delta t}{2 \varepsilon \kappa_y + \sigma_y \Delta t} \]

\[ C_{2Bx} = \frac{2 \varepsilon \Delta t}{2 \varepsilon \kappa_y \Delta y + \sigma_y \Delta t \Delta y} \]

$B_x(i_0 \leq i \leq i_1, j_0 - 0.5)$

\[ B_x^{n+1}_{i+0.5,j_0-0.5} = C_{1Bx} B_x^{n+1}_{i+0.5,j_0-0.5} - C_{2Bx} [E_z^{n+1}_{i+0.5,j_0-0.5} - E_z^{n+1}_{i-0.5,j_0-1}] + C_{2Bx} E_{z,inc}^{n+1}_{i+0.5,j_0-0.5} \]

$B_x(i_0 \leq i \leq i_1, j_1 + 0.5)$

\[ B_x^{n+1}_{i+0.5,j_1+0.5} = C_{1Bx} B_x^{n+1}_{i+0.5,j_1+0.5} - C_{2Bx} [E_z^{n+1}_{i+0.5,j_1+0.5} - E_z^{n+1}_{i-0.5,j_1+1}] - C_{2Bx} E_{z,inc}^{n+1}_{i+0.5,j_1+0.5} \]

3. $E_z, D_z$ consistency conditions

\[ D_z^{n+0.5,0.5}_{i+0.5,j+0.5} = C_{1Dz} D_z^{n+0.5,0.5}_{i+0.5,j+0.5} + C_{2Dz} [H_y^{n+1}_{i,j+0.5} - H_y^{n}_{i,j+0.5}] - C_{3Dz} [H_x^{n+1}_{i+0.5,j+0.5} - H_x^{n}_{i+0.5,j+0.5}] \]
\[ D_x(i_0, j_0 \leq j \leq j_1) \]
\[ D_x^{n+0.5,j+0.5} = (D_x^{n+0.5,j+0.5}) = C_{2Dz} \left[ H_y,inc \right]_{i,j+0.5}^n \]

\[ D_x(i_1, j_0 \leq j \leq j_1) \]
\[ D_x^{n+0.5,j+0.5} = (D_x^{n+0.5,j+0.5}) + C_{2Dz} \left[ H_y,inc \right]_{i+1,j+0.5}^n \]

\[ D_x(i_0 \leq i \leq i_1, j_0) \]
\[ D_x^{n+0.5,j+0.5} = (D_x^{n+0.5,j+0.5}) + C_{3Dz} \left[ H_x,inc \right]_{i+0.5,j}^n \]

\[ D_x(i_0 \leq i \leq i_1, j_1) \]
\[ D_x^{n+0.5,j+0.5} = (D_x^{n+0.5,j+0.5}) - C_{3Dz} \left[ H_x,inc \right]_{i+0.5,j+1}^n \]
APPENDIX E: 2D FDTD – UPML code with TF / SF formulation

% 2D FDTD + UPML + TF / SF
% S. Sandeep, 2009.
% Notes :: E at half integer time intervals and H at integer time
% intervals.
%---------------------------------------------------------------------
% Fundamental constants
f = 1e9;
c = 3e8;
mu_o = 4*pi*1e-7;
eps_o = 8.85e-12;

% Dimension of the problem space
ROWS_AIR = 351;
COLS_AIR = 351;

% Dimension of the computational space
PML_ls = 16;
ROWMAX = (2*PML_ls) + ROWS_AIR;
COLMAX = (2*PML_ls) + COLS_AIR;

dx = 0.03 / 6;
dy = 0.03 / 6;
dt = dx / (2*c);
tH = 0; % Time index for H
tE = 0; % Time index for E
time = 0; % Current time in units of dt
TimeDur = 2000;
kz = 1;
sigmaz = 0;
g = 1.3;
const = g ^ (1/dx);
sigma0 = 1e-2;

EzData = zeros(ROWMAX,COLMAX,TimeDur - 1);

%---------------------------------------------------------------------
% TF / SF settings
% TF / SF boundary location
% TFSFBndryOffset : Fraction of ROWS_AIR / COLS_AIR. Offset from problem
% space - UPML boundary
TFSFBndryOffset = 0.3;
offset = ROWS_AIR * TFSFBndryOffset;
if(offset - round(offset) > 0)
  i0 = round(offset) + 0.5;
else
  i0 = round(offset) - 0.5;
end
% (i0,j0) is the left - bottom point of the TF / SF region
% il - i0 = dimension of TF / SF region
il = PML_ls + i0 + (ROWS_AIR - 2 * i0);
i0 = PML_ls + i0;
j0 = i0;
j1 = i1;

% Wave angle
phi = 90 * pi / 180;

% Incident wave vector (unit vector)
kinc = [cos(phi) ; sin(phi)];

% Select the TF / SF boundary coordinate that the incident wave first strikes : (iw,jw)
phideg = phi * 180 / pi;
if (phideg > 0 && phideg <= 90)
iw = i0;
jw = j0;
elseif (phideg >= 90 && phideg < 180)
iw = i1;
jw = j0;
elseif (phideg >= 180 && phideg < 270)
iw = i1;
jw = j1;
else
iw = i0;
jw = j1;
end

% Incident wave generation
% Use d = dx, dt = dt from the 2D FDTD for the auxiliary one dimensional incident wave FDTD
d = dx;
% Courant stability criterion
S = c * dt / d;
Nl = (c / f) / d;
vp_phi = NumPhaseVel(phi,d,S,Nl,0.1);
vp_o = NumPhaseVel(0,d,S,Nl,0.1);
vp_rat = vp_o / vp_phi;

% The direction of the auxiliary 1D FDTD axis depends on wave angle. Its length is maximum when wave angle = 45 degrees. For a square TF region. The number of E points on the diagonal = number of points on on side of TF / SF boundary. Add 4 to this, 2 on either end
AuxFDTDSize = ceil( (i1 - i0) * 1.414 ) + 1 + 4;

% At time = 0, (tH = 1,tE = 0)
Einc = zeros(1,AuxFDTDSize);
Hinc = zeros(1,AuxFDTDSize);
Hincbc1 = 0;
Hincbc2 = 0;

% Constitute parameter matrices
sigmax_Hx = zeros(ROWMAX + 1,COLMAX);
kx_Hx = ones(ROWMAX + 1,COLMAX);
sigmay_Bx = zeros(ROWMAX + 1,COLMAX);
ky_Bx = ones(ROWMAX + 1,COLMAX);
ky_Hy = ones(ROWMAX,COLMAX + 1);
sigmay_Hy = zeros(ROWMAX,COLMAX + 1);
kx_Hy = ones(ROWMAX,COLMAX + 1);
sigmax_Hy = zeros(ROWMAX,COLMAX + 1);
kx_Dz = ones(ROWMAX,COLMAX);
sigmax_Dz = zeros(ROWMAX,COLMAX);
ky_Ez = ones(ROWMAX,COLMAX);
sigmay_Ez = zeros(ROWMAX,COLMAX);

epsr = ones (ROWMAX,COLMAX);
sigma = zeros(ROWMAX,COLMAX);

ROWCNT = (ROWMAX - 1) / 2;
COLCNT = (COLMAX - 1) / 2;
for row = 1 : ROWMAX
    for col = 1 : COLMAX
        sigma(row,col) = 0;
        epsr(row,col) = 1;
        % Square cylinder
        if(row > (ROWMAX / 2) - (ROWMAX / 20) && row < (ROWMAX / 2) + (ROWMAX / 20))
            if(col > (COLMAX / 2) - (COLMAX / 20) && col < (COLMAX / 2) + (COLMAX / 20))
                sigma(row,col) = 1e6;
            end
        end
        % Circular cylinder
        if(norm([row - ROWCNT,col - COLCNT]) <= 20)
            sigma(row,col) = 1e6;
        end
    end
end

% Calculate parameter matrices
% kx_Hx,sigma_Hx,ky_Bx,sigma_Bx,ky_Hy,sigma_Hy,kx_Hy,sigmax_Hy,kx_Dz,
% sigmax_Dz,ky_Ez,sigmay_Ez
% Calculate kx_Hx and sigma_Hx
for row = 1 : ROWMAX + 1
    for col = 1 : COLMAX
        if col > PML_ls && col <= COLMAX - PML_ls
            sigmax_Hx(row,col) = 0;
            kx_Hx(row,col) = 1;
            % Inside the problem space
            if(row >= PML_ls + 1 && row <= PML_ls + ROWS_AIR + 1)
                % Interface condition
                sigmax_Hx(row,col) = ( sigma(row,col) + sigma(row - 1,col) ) / 2;
            end
        elseif col <= PML_ls
            x = (PML_ls - col) * dx + dx/2;
            sigmax_Hx(row,col) = (const^x)*sigma0;
            kx_Hx(row,col) = const^x;
        else
            x = (col - PML_ls - COLS_AIR - 1) * dx + dx/2;
            sigmax_Hx(row,col) = (const^x)*sigma0;
            kx_Hx(row,col) = const^x;
%Calculate ky_Bx and sigmay_Bx
for row = 1 : ROWMAX + 1
   for col = 1 : COLMAX
      if row <= PML_ls + 1
         y = (PML_ls + 1 - row) * dy;
         sigmay_Bx(row,col) = (const^y)*sigma0;
         ky_Bx(row,col) = const^y;
      elseif row >= PML_ls + ROWS_AIR + 1
         y = (row - PML_ls - ROWS_AIR - 1) * dy;
         sigmay_Bx(row,col) = (const^y)*sigma0;
         ky_Bx(row,col) = const^y;
      else
         sigmay_Bx(row,col) = 0;
         ky_Bx(row,col) = 1;
      end
   end
end
%Calculate ky_Hy and sigmay_Hy
for row = 1 : ROWMAX
   for col = 1 : COLMAX + 1
      if row <= PML_ls
         y = (PML_ls - row)*dy + dy/2;
         sigmay_Hy(row,col) = (const^y)*sigma0;
         ky_Hy(row,col) = const^y;
      elseif row >= PML_ls + ROWS_AIR + 1
         y = (row - PML_ls - ROWS_AIR - 1)*dy + dy/2;
         sigmay_Hy(row,col) = (const^y)*sigma0;
         ky_Hy(row,col) = const^y;
      else
         sigmay_Hy(row,col) = 0;
         ky_Hy(row,col) = 1;
      end
   end
end
%Calculate kx_Hy and sigmax_Hy
for row = 1 : ROWMAX
   for col = 1 : COLMAX + 1
      if col <= PML_ls + 1
         x = (PML_ls + 1 - col) * dx;
         sigmax_Hy(row,col) = (const^x)*sigma0;
         kx_Hy(row,col) = const^x;
      elseif col >= PML_ls + COLS_AIR + 1
         x = (col - PML_ls - COLS_AIR - 1)*dx + dx/2;
         sigmax_Hy(row,col) = (const^x)*sigma0;
         kx_Hy(row,col) = const^x;
      else
         sigmax_Hy(row,col) = 0;
         kx_Hy(row,col) = 1;
      end
   end
end
\begin{verbatim}
x = (col - PML_ls - COLS_AIR - 1) * dx;
sigmax_Hy(row, col) = (const^x) * sigma0;
kx_Hy(row, col) = const^x;
else
  sigmax_Hy(row, col) = 0;
kx_Hy(row, col) = 1;
% Inside the problem space
if (row >= PML_ls + 1 && row <= PML_ls + ROWS_AIR)
  % Interface condition
  sigmax_Hy(row, col) = (sigma(row, col) + sigma(row, col - 1)) / 2;
else
end
end

% Calculate the kx_Dz and sigmax_Dz for row = 1 : ROWMAX
for col = 1 : COLMAX
  if col > PML_ls && col <= COLMAX - PML_ls
    sigmax_Dz(row, col) = 0;
kx_Dz(row, col) = 1;
  else
    sigmax_Dz(row, col) = (const^x) * sigma0;
kx_Dz(row, col) = const^x;
  end
end

% Calculate ky_Ez and sigmay_Ez for row = 1 : ROWMAX
for col = 1 : COLMAX
  if row <= PML_ls
    y = (PML_ls - row) * dy + dy/2;
sigmay_Ez(row, col) = (const^y) * sigma0;
kx_Ez(row, col) = const^y;
  elseif row >= PML_ls + ROWS_AIR + 1
    y = (row - PML_ls - ROWS_AIR - 1) * dy + dy/2;
sigmay_Ez(row, col) = (const^y) * sigma0;
kx_Ez(row, col) = const^y;
  else
    sigmay_Ez(row, col) = 0;
kx_Ez(row, col) = 1;
  end
end
end
\end{verbatim}
% Iteration of update equations
% Bx_1 -> previous Bx (i.e. n - 1)
% Bx_2 -> current Bx (i.e. n)

% Initialisation of Hx, Hy, Bx, By, Dz, Ez at time = 0 (tH = 1, tE = 0)
% time = 0;
tH = 1;
Bx_1 = zeros(ROWMAX+1,COLMAX);
Hx_1 = zeros(ROWMAX+1,COLMAX);
By_1 = zeros(ROWMAX,COLMAX+1);
Hy_1 = zeros(ROWMAX,COLMAX+1);
Ez_1 = zeros(ROWMAX,COLMAX);
Dz_1 = zeros(ROWMAX,COLMAX);

Bx_2 = zeros(ROWMAX+1,COLMAX);
Hx_2 = zeros(ROWMAX+1,COLMAX);
By_2 = zeros(ROWMAX,COLMAX+1);
Hy_2 = zeros(ROWMAX,COLMAX+1);
Ez_2 = zeros(ROWMAX,COLMAX);
Dz_2 = zeros(ROWMAX,COLMAX);

% Excitation of Ez at time = 0.5 (tE = 1)
% time = 0.5;
tE   = 1;
Einc(1,1) = exp(-(((time * dt - 60 * dt) / (20 * dt)) ^ 2));

while time < TimeDur
    time = time + 0.5;
tH = tH + 1;

    % Calculate Bx
    % Bx|n require Einc|n - 0.5
    for row = 1 : ROWMAX + 1
        for col = 1 : COLMAX
            epsr_Bx = eps;
            C1 = (2*eps_o*ky_Bx(row,col) -
                 sigmay_Bx(row,col)*dt)/(2*eps_o*ky_Bx(row,col) + sigmay_Bx(row,col)*dt);
            C2 = 2*eps_o*dt/(2*eps_o*ky_Bx(row,col) +
                 sigmay_Bx(row,col)*dt);
            if row == 1
                dEz = -1*Ez_1(row,col);
            elseif row == ROWMAX + 1
                dEz = Ez_1(row-1,col) - 0;
            else
                dEz = Ez_1(row - 1,col) - Ez_1(row,col);
            end
            cc1 = 0;
            cc2 = 0;
        end
    end
end

EzData(:, :, tE) = Ez_2;
% Consistency condition
if (row == ROWMAX - j1 + 0.5 && col >= i0 + 0.5 && col <= i1 + 0.5)
    ip = col - 0.5;
    jp = j1;

    rp = [ip - iw ; jp - jw];
    D = kinc' * rp;
    m0 = 3 + floor(D);
    m1 = m0 + 1;

    E0 = Einc(1,m0);
    E1 = Einc(1,m1);
    Ep = (E0 - E1) * (D - floor(D)) + E0;
    cc1 = - (C2 / dy) * Ep;
elseif (row == ROWMAX - j0 + 1.5 && col >= i0 + 0.5 && col <= i1 + 0.5)
    ip = col - 0.5;
    jp = j0;

    rp = [ip - iw ; jp - jw];
    D = kinc' * rp;
    m0 = 3 + floor(D);
    m1 = m0 + 1;

    E0 = Einc(1,m0);
    E1 = Einc(1,m1);
    Ep = (E0 - E1) * (D - floor(D)) + E0;
    cc2 = + (C2 / dy) * Ep;
end

Bx_2(row,col) = C1 * Bx_1(row,col) - (C2/dy) * (dEz) + cc1 + cc2;
end
end

%Calculate Hx
for row = 1:ROWMAX + 1
    for col = 1:COLMAX
        C1 = (2*eps_o*kx_Hx(row,col) +
             sigmax_Hx(row,col)*dt)/(2*eps_o*mu_o*kz + mu_o*sigmaz*dt);
        C2 = (sigmax_Hx(row,col)*dt -
             2*eps_o*kx_Hx(row,col))/(2*eps_o*mu_o*kz + sigmaz*dt);
        C3 = (sigmaz*dt - 2*kz*eps_o)/(2*kz*eps_o + sigmaz*dt);
        Hx_2(row,col) = C1 * Bx_2(row,col) + C2 * Bx_1(row,col) - (C3 *
                      Hx_1(row,col));
    end
end

%Calculate By
for row = 1 : ROWMAX
    for col = 1 : COLMAX + 1
        C1 = (2*eps_o*kz - sigmaz*dt)/(2*eps_o*kz + sigmaz*dt);
        C2 = 2*eps_o*dt/(2*eps_o*kz + sigmaz*dt);
        if col == 1
\[ \text{dE}_z = \text{Ez}_1(\text{row}, \text{col}); \]
\[
\text{elseif } \text{col} == \text{COLMAX} + 1 \\
\quad \text{dE}_z = -1 \times \text{Ez}_1(\text{row}, \text{col} - 1); \]
\[
\text{else} \\
\quad \text{dE}_z = \text{Ez}_1(\text{row}, \text{col}) - \text{Ez}_1(\text{row}, \text{col} - 1); \]
\end{verbatim}

\[
\text{cc1} = 0; \]
\[
\text{cc2} = 0; \]
\[
% \text{Consistency conditions} \]
\[
% \text{col} = i_0 + 0.5 \text{ corresponds to } B_x / H_x \text{ at } i_0 - 0.5 \]
\[
\text{if } (\text{col} == i_0 + 0.5 \text{ && } \text{row} >= \text{ROWMAX} - j_1 + 0.5 \text{ && } \text{row} <= \text{ROWMAX} - j_0 + 0.5) \]
\[
\text{ip} = i_0; \\
\text{jp} = \text{ROWMAX} - \text{row} + 0.5; \\
\text{rp} = \text{[ip - iw; jp - jw]}; \\
\text{D} = \text{kinc}' \times \text{rp}; \\
\text{m0} = 3 + \text{floor}(\text{D}); \\
\text{ml} = \text{m0} + 1; \\
\text{E0} = \text{Einc}(1, \text{m0}); \\
\text{E1} = \text{Einc}(1, \text{ml}); \\
\text{Ep} = (\text{E0} - \text{E1}) \times (\text{D} - \text{floor}(\text{D})) + \text{E0}; \\
\text{cc1} = - (\text{C2} / \text{dx}) \times \text{Ep}; \\
\text{elseif } (\text{col} == i_1 + 1.5 \text{ && } \text{row} >= \text{ROWMAX} - j_1 + 1.5 \text{ && } \text{row} <= \text{ROWMAX} - j_0 + 0.5) \\
\text{ip} = i_1; \\
\text{jp} = \text{ROWMAX} - \text{row} + 0.5; \\
\text{rp} = \text{[ip - iw; jp - jw]}; \\
\text{D} = \text{kinc}' \times \text{rp}; \\
\text{m0} = 3 + \text{floor}(\text{D}); \\
\text{ml} = \text{m0} + 1; \\
\text{E0} = \text{Einc}(1, \text{m0}); \\
\text{E1} = \text{Einc}(1, \text{ml}); \\
\text{Ep} = (\text{E0} - \text{E1}) \times (\text{D} - \text{floor}(\text{D})) + \text{E0}; \\
\text{cc2} = + (\text{C2} / \text{dx}) \times \text{Ep}; \]
\end{verbatim}

\[
\text{By}_2(\text{row}, \text{col}) = \text{C1} \times \text{By}_1(\text{row}, \text{col}) + (\text{C2/dx}) \times \text{dE}_z + \text{cc1} + \text{cc2}; \]
\]
\end{verbatim}

%Calculate Hy
\[
\text{for row} = 1 : \text{ROWMAX} \\
\quad \text{for col} = 1 : \text{COLMAX} + 1 \\
\quad \quad \text{C1} = (2 \times \text{eps}_o \times \text{ky}_Hy(\text{row}, \text{col}) + \text{sigmay}_Hy(\text{row}, \text{col}) \times \text{dt}) / (2 \times \text{eps}_o \times \text{mu}_o \times \text{kx}_Hy(\text{row}, \text{col}) + \text{sigmax}_Hy(\text{row}, \text{col}) \times \text{mu}_o \times \text{dt}); \\
\end{verbatim}
C2 = (sigmay_Hy(row,col)*dt - 2*eps_o*ky_Hy(row,col))/(sigmax_Hy(row,col)*mu_o*dt + 2*eps_o*mu_o*kk_Hy(row,col));
C3 = (sigmax_Hy(row,col)*dt - 2*eps_o*kk_Hy(row,col))/(2*eps_o*kk_Hy(row,col) + sigmax_Hy(row,col)*dt);
Hy_2(row,col) = C1*By_2(row,col) + C2*By_1(row,col) - C3*Hy_1(row,col);

% Update Hinc at time = n (tH = n + 1)
for i = 1 : AuxFDTDSize - 1
    Hinc(1,i) = Hinc(1,i) + ((dt / vp_rat) / (mu_o * d )) * (Einc(1,i) - Einc(1,i + 1));
end

% Analytical absorbing boundary condition at the end of the auxiliary
% 1D incident field FDTD
if (tH <= 2)
    Hinc(1,AuxFDTDSize) = 0;
    if(tH == 1)
        Hincbc1 = Hinc(1,AuxFDTDSize - 1);
        Hincbc2 = Hinc(1,AuxFDTDSize - 1);
    end
else
    if(mod(tH,2))
        Hinc(1,AuxFDTDSize) = Hincbc1;
        Hincbc1 = Hinc(1,AuxFDTDSize - 1);
    else
        Hinc(1,AuxFDTDSize) = Hincbc2;
        Hincbc2 = Hinc(1,AuxFDTDSize - 1);
    end
end

% time = n + 0.5, tE = n + 1
    time = time  + 0.5;
    tE = tE + 1;
%Calculate Dz
for row = 1:ROWMAX
    for col = 1:COLMAX
        C1 = 2*eps_o*dt/(2*eps_o*kk_Dz(row,col) +sigmax_Dz(row,col)*dt);
        C2 = (2*eps_o*kk_Dz(row,col) -
        sigmax_Dz(row,col)*dt)/(2*eps_o*kk_Dz(row,col) + sigmax_Dz(row,col)*dt);
        dHy = Hy_2(row,col+1) - Hy_2(row,col);
        dHx = Hx_2(row,col) - Hx_2(row+1,col);
        Dz_2(row,col) = (C1/dy)*dHy - (C1/dx)*dHx + C2*Dz_1(row,col);
    end
end
% Consistency condition
cc1 = 0;
cc2 = 0;
cc3 = 0;
cc4 = 0;
if(col == i0 + 0.5 && row >= ROWMAX - j1 + 0.5 && row <= ROWMAX - j0 + 0.5)
    ip = i0 - 0.5;
\begin{align*}
\text{jp} &= \text{ROWMAX} - \text{row} + 0.5; \\
\text{rp} &= [\text{ip} - \text{iw} ; \text{jp} - \text{jw}]; \quad \text{D} = \text{kinc}' \cdot \text{rp}; \\
\text{m0} &= 1 + \text{floor(D + 1.5)}; \\
\text{m1} &= \text{m0} + 1; \\
\text{H0} &= \text{Hinc}(1, \text{m0}); \\
\text{H1} &= \text{Hinc}(1, \text{m1}); \\
\text{Hp} &= (\text{H0} - \text{H1}) \cdot (\text{D} + 1.5 - \text{floor(D + 1.5)}) + \text{H0}; \\
\text{cc1} &= -(\text{C1} / \text{dx}) \cdot \text{Hp} \cdot (-\cos(\phi)); \\
\text{elseif} (\text{col} == \text{i1} + 0.5 \quad \&\& \text{row} >= \text{ROWMAX} - \text{j1} + 0.5 \quad \&\& \text{row} <= \text{ROWMAX} - \text{j0} + 0.5) \\
\text{ip} &= \text{i1} + 1.5; \\
\text{jp} &= \text{ROWMAX} - \text{row} + 0.5; \\
\text{rp} &= [\text{ip} - \text{iw} ; \text{jp} - \text{jw}]; \quad \text{D} = \text{kinc}' \cdot \text{rp}; \\
\text{m0} &= 1 + \text{floor(D + 1.5)}; \\
\text{m1} &= \text{m0} + 1; \\
\text{H0} &= \text{Hinc}(1, \text{m0}); \\
\text{H1} &= \text{Hinc}(1, \text{m1}); \\
\text{Hp} &= (\text{H0} - \text{H1}) \cdot (\text{D} + 1.5 - \text{floor(D + 1.5)}) + \text{H0}; \\
\text{cc2} &= + (\text{C1} / \text{dx}) \cdot \text{Hp} \cdot (-\cos(\phi)); \\
\text{elseif} (\text{row} == \text{ROWMAX} - \text{j1} + 0.5 \quad \&\& \text{col} >= \text{i0} + 0.5 \quad \&\& \text{col} <= \text{i1} + 0.5) \\
\text{ip} &= \text{col} - 0.5; \\
\text{jp} &= \text{j1} + 0.5; \\
\text{rp} &= [\text{ip} - \text{iw} ; \text{jp} - \text{jw}]; \quad \text{D} = \text{kinc}' \cdot \text{rp}; \\
\text{m0} &= 1 + \text{floor(D + 1.5)}; \\
\text{m1} &= \text{m0} + 1; \\
\text{H0} &= \text{Hinc}(1, \text{m0}); \\
\text{H1} &= \text{Hinc}(1, \text{m1}); \\
\text{Hp} &= (\text{H0} - \text{H1}) \cdot (\text{D} + 1.5 - \text{floor(D + 1.5)}) + \text{H0}; \\
\text{cc3} &= -(\text{C1} / \text{dy}) \cdot \text{Hp} \cdot (\sin(\phi)); \\
\text{elseif} (\text{row} == \text{ROWMAX} - \text{j0} + 0.5 \quad \&\& \text{col} >= \text{i0} + 0.5 \quad \&\& \text{col} <= \text{i1} + 0.5) \\
\text{ip} &= \text{col} - 0.5; \\
\text{jp} &= \text{j0} - 0.5; \\
\text{rp} &= [\text{ip} - \text{iw} ; \text{jp} - \text{jw}]; \quad \text{D} = \text{kinc}' \cdot \text{rp}; \\
\text{m0} &= 1 + \text{floor(D + 1.5)}; \\
\text{m1} &= \text{m0} + 1; \\
\text{H0} &= \text{Hinc}(1, \text{m0}); \\
\text{H1} &= \text{Hinc}(1, \text{m1}); \\
\text{Hp} &= (\text{H0} - \text{H1}) \cdot (\text{D} + 1.5 - \text{floor(D + 1.5)}) + \text{H0};
\end{align*}
cc3 = + (C1 / dy) * Hp * (sin(phi));
end
Dz_2(row,col) = Dz_2(row,col) + cc1 + cc2 + cc3 + cc4;
end
end
%Calculate Ez
for row = 1 : ROWMAX
    for col = 1:COLMAX
        C1 = (2*eps_o*ky_Ez(row,col) - sigmay_Ez(row,col)*dt)/(2*eps_o*ky_Ez(row,col) + sigmay_Ez(row,col)*dt);
        C2 = (2*eps_o*kz + sigmaz*dt)/(2*eps_o*ky_Ez(row,col) + sigmay_Ez(row,col)*dt);
        C3 = (sigmaz*dt - 2*eps_o*kz )/(2*eps_o*ky_Ez(row,col) + sigmay_Ez(row,col)*dt);
        Ez_2(row,col) = C1*Ez_1(row,col) + (C2/eps_o)*Dz_2(row,col) + (C3/eps_o)*Dz_1(row,col);
    end
end
% Store Ez values at all time
EzData(:,:,tE) = Ez_2;
% Update Einc at time = n + 0.5 (tE = n + 1)
for i = 2 : AuxFDTDSize
    Einc(1,i) = Einc(1,i) + ((dt / vp_rat) / (eps_o * d)) * (Hinc(1,i - 1) - Hinc(1,i));
end
% Incient wave excitation
Einc(1,1) = sin(2 * pi * 1e9 * time * dt);%exp(-(((time * dt - 60 * dt) / (10 * dt)) ^ 2));
Ez_1 = Ez_2;
Bx_1 = Bx_2;
Hx_1 = Hx_2;
By_1 = By_2;
Hy_1 = Hy_2;
Dz_1 = Dz_2;
end