Progress In Electromagnetics Research M, Vol. 14, 177–192, 2010

GPU-ACCELERATED FUNDAMENTAL ADI-FDTD WITH COMPLEX FREQUENCY SHIFTED CONVOLUTIONAL PERFECTLY MATCHED LAYER

W. C. Tay, D. Y. Heh, and E. L. Tan

School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798, Singapore

Abstract—This paper presents the graphics processing unit (GPU) accelerated fundamental alternating-direction-implicit finite-difference time-domain (FADI-FDTD) with complex frequency shifted convolutional perfectly matched layer (CFS-CPML). The compact matrix form of the conventional ADI-FDTD method with CFS-CPML is formulated into FADI-FDTD with its right-hand-sides free of matrix operators, resulting in simpler and more concise update equations. Using Compute Unified Device Architecture (CUDA), the FADI-FDTD with CFS-CPML is further incorporated into the GPU to exploit data parallelism. Numerical results show that a much higher efficiency gain of up to 15 times can be achieved.

1. INTRODUCTION

The alternating-direction-implicit finite-difference time-domain (ADI-FDTD) method [1, 2] has been widely used due to its unconditionally stable feature where the time step size is unrestricted by the Courant-Friedrichs-Lewy (CFL) stability condition. However, this comes at an expense of increasing complexity in its implementation. Besides having to solve the tridiagonal systems, there are substantial amount of arithmetic operations and field variables involved on the right-hand-sides of the update equations, not to mention the huge amount of memory indexing operations incurred. Moreover, to solve open structure problems, it is still necessary to employ absorbing boundary

Received 6 September 2010, Accepted 12 October 2010, Scheduled 21 October 2010
Corresponding author: Wei Choon Tay (tayw0034@e.ntu.edu.sg).

conditions (ABCs) [3–5], which involves more of these overheads. Among the ABCs, the perfectly matched layer (PML) [5] remains popular due to its effectiveness in absorbing outgoing electromagnetic waves. In [6], the convolutional perfectly matched layer (CPML) with complex frequency shifted (CFS) parameters has been introduced for the unconditionally stable ADI-FDTD method. By implementing CFS-CPML into the ADI-FDTD algorithm, the already rather complex ADI update equations becomes even more complicated. With the inclusion of CFS-CPML parameters, the amount of aforementioned field variables, arithmetic and memory indexing operations have all increased considerably. This results in further degradation of the overall efficiency, which is undesirable.

Recently, an efficient algorithm has been developed for the ADI-FDTD [7]. Such algorithm is included within a family of fundamental implicit schemes, which feature similar fundamental updating structures that are in simplest forms with most efficient matrix-operator-free right-hand-sides [8]. This leads to fundamental ADI-FDTD, or FADI-FDTD in short, which results in much simpler and more concise update equations than the conventional ADI-FDTD implementation. Nevertheless, despite having a more efficient and simpler implementation using the FADI-FDTD, continuing efforts are still being made to further increase the overall efficiency. Of late, programmable graphics processing units (GPUs) with highly parallel processors have led to the interest in using GPUs for general purpose programming [9–12]. Such highly parallel processing feature of the GPU further motivates us into exploring the implementation of the FADI-FDTD.

In this paper, we shall present the GPU-accelerated FADI-FDTD with CFS-CPML. The conventional ADI-FDTD method with CFS-CPML is first cast into the compact matrix form. Based on [8], the matrix form is formulated into FADI-FDTD with its right-hand-sides free of matrix operators, resulting in simpler and more concise update equations. Using Compute Unified Device Architecture (CUDA), we further incorporate the FADI-FDTD with CFS-CPML into the GPU to exploit data parallelism. Numerical results of reflection error and efficiency gain will be presented.

2. FADI-FDTD WITH CFS-CPML

The PML medium is an artificial region used to absorb outgoing electromagnetic waves for truncating computational domain. Within this region, the Maxwell's equations are expressed in the stretched

coordinate space as

$$j\omega\epsilon\mathbf{E} = \tilde{\boldsymbol{\nabla}} \times \mathbf{H} \tag{1a}$$

$$-j\omega\mu\mathbf{H} = \tilde{\boldsymbol{\nabla}} \times \mathbf{E} \tag{1b}$$

where

$$\tilde{\boldsymbol{\nabla}} = \hat{\boldsymbol{x}}\frac{1}{s_x}\frac{\partial}{\partial x} + \hat{\boldsymbol{y}}\frac{1}{s_y}\frac{\partial}{\partial y} + \hat{\boldsymbol{z}}\frac{1}{s_z}\frac{\partial}{\partial z}, \tag{2}$$

and the stretched coordinate metric is given as

$$s_\zeta = \kappa_\zeta + \frac{\sigma_\zeta}{\alpha_\zeta + j\omega\epsilon}, \quad \zeta = x, y, z. \tag{3}$$

We first write the conventional ADI-FDTD method with CFS-CPML [6] in compact matrix form as

$$\left(\mathbf{I} - \frac{\Delta t}{2}\left(\mathbf{A}' + \frac{\mathbf{L}}{2}\right)\right)\mathbf{u}^{n+\frac{1}{2}} = \left(\mathbf{I} + \frac{\Delta t}{2}\left(\mathbf{B}' + \frac{\mathbf{L}}{2}\right)\right)\mathbf{u}^{n} + \frac{\Delta t}{2}\mathbf{W}\Psi^{n} \tag{4a}$$

$$\Psi^{n+\frac{1}{2}} = \mathbf{C}\Psi^{n} + \mathbf{D}\mathbf{u}^{n+\frac{1}{2}} \tag{4b}$$

$$\left(\mathbf{I} - \frac{\Delta t}{2}\left(\mathbf{B}' + \frac{\mathbf{L}}{2}\right)\right)\mathbf{u}^{n+1} = \left(\mathbf{I} + \frac{\Delta t}{2}\left(\mathbf{A}' + \frac{\mathbf{L}}{2}\right)\right)\mathbf{u}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\mathbf{W}\Psi^{n+\frac{1}{2}} \tag{4c}$$

$$\Psi^{n+1} = \mathbf{C}\Psi^{n+\frac{1}{2}} + \mathbf{D}\mathbf{u}^{n+1} \tag{4d}$$

where

$$\mathbf{u} = [E_x \quad E_y \quad E_z \quad H_x \quad H_y \quad H_z]^T,$$

$$\mathbf{A}' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{1}{\epsilon\kappa_y}\frac{\partial}{\partial y} \\ 0 & 0 & 0 & \frac{1}{\epsilon\kappa_z}\frac{\partial}{\partial z} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\epsilon\kappa_x}\frac{\partial}{\partial x} & 0 \\ 0 & \frac{1}{\mu\kappa_z}\frac{\partial}{\partial z} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\mu\kappa_x}\frac{\partial}{\partial x} & 0 & 0 & 0 \\ \frac{1}{\mu\kappa_y}\frac{\partial}{\partial y} & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{B}' = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{-1}{\epsilon\kappa_z}\frac{\partial}{\partial z} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-1}{\epsilon\kappa_x}\frac{\partial}{\partial x} \\ 0 & 0 & 0 & \frac{-1}{\epsilon\kappa_y}\frac{\partial}{\partial y} & 0 & 0 \\ 0 & 0 & \frac{-1}{\mu\kappa_y}\frac{\partial}{\partial y} & 0 & 0 & 0 \\ \frac{-1}{\mu\kappa_z}\frac{\partial}{\partial z} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-1}{\mu\kappa_x}\frac{\partial}{\partial x} & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{L} = \begin{bmatrix} -\frac{\sigma}{\epsilon} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{\sigma}{\epsilon} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{\sigma}{\epsilon} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{\sigma^*}{\mu} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{\sigma^*}{\mu} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{\sigma^*}{\mu} \end{bmatrix},$$

$$\mathbf{W} = \begin{bmatrix} \frac{1}{\epsilon}\Theta & \mathbf{O}_{3\times 6} \\ \mathbf{O}_{3\times 6} & \frac{1}{\mu}\Theta \end{bmatrix}, \quad \Theta = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{C_e} & \mathbf{O}_{6\times 6} \\ \mathbf{O}_{6\times 6} & \mathbf{C_h} \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} \mathbf{O}_{6\times 3} & \mathbf{D_e} \\ \mathbf{D_h} & \mathbf{O}_{6\times 3} \end{bmatrix}, \quad \Psi = \begin{bmatrix} \Psi_E & \Psi_H \end{bmatrix}^T,$$

$$\mathbf{C_e} = \begin{bmatrix} c_y & 0 & 0 & 0 & 0 & 0 \\ 0 & c_z & 0 & 0 & 0 & 0 \\ 0 & 0 & c_z & 0 & 0 & 0 \\ 0 & 0 & 0 & c_x & 0 & 0 \\ 0 & 0 & 0 & 0 & c_x & 0 \\ 0 & 0 & 0 & 0 & 0 & c_y \end{bmatrix}, \quad \mathbf{D_e} = \begin{bmatrix} 0 & 0 & d_y\frac{\partial}{\partial y} \\ 0 & d_z\frac{\partial}{\partial z} & 0 \\ d_z\frac{\partial}{\partial z} & 0 & 0 \\ 0 & 0 & d_x\frac{\partial}{\partial x} \\ 0 & d_x\frac{\partial}{\partial x} & 0 \\ d_y\frac{\partial}{\partial y} & 0 & 0 \end{bmatrix},$$

$$\mathbf{C_h} = \begin{bmatrix} c_z & 0 & 0 & 0 & 0 & 0 \\ 0 & c_y & 0 & 0 & 0 & 0 \\ 0 & 0 & c_x & 0 & 0 & 0 \\ 0 & 0 & 0 & c_z & 0 & 0 \\ 0 & 0 & 0 & 0 & c_y & 0 \\ 0 & 0 & 0 & 0 & 0 & c_x \end{bmatrix}, \quad \mathbf{D_h} = \begin{bmatrix} 0 & d_z\frac{\partial}{\partial z} & 0 \\ 0 & 0 & d_y\frac{\partial}{\partial y} \\ 0 & 0 & d_x\frac{\partial}{\partial x} \\ d_z\frac{\partial}{\partial z} & 0 & 0 \\ d_y\frac{\partial}{\partial y} & 0 & 0 \\ 0 & d_x\frac{\partial}{\partial x} & 0 \end{bmatrix},$$

$$\Psi_E = \begin{bmatrix} \psi_{e_{xy}} & \psi_{e_{xz}} & \psi_{e_{yz}} & \psi_{e_{yx}} & \psi_{e_{zx}} & \psi_{e_{zy}} \end{bmatrix},$$

$$\Psi_H = \begin{bmatrix} \psi_{h_{xz}} & \psi_{h_{xy}} & \psi_{h_{yx}} & \psi_{h_{yz}} & \psi_{h_{zy}} & \psi_{h_{zx}} \end{bmatrix}.$$

$\mathbf{O}_{p\times q}$ is the null matrix with $p \times q$ dimension, Δt is the time step, σ and σ^* are the electric and magnetic conductivities, respectively. Note that σ is different from σ_ζ in (3), with the latter referring to the conductivity profile of the PML. c_ζ and d_ζ are the update coefficients for Ψ in the PML regions defined as

$$c_\zeta = e^{-(\frac{\sigma_\zeta}{\kappa_\zeta}+\alpha_\zeta)\frac{\Delta t}{2\epsilon}}, \quad d_\zeta = \frac{\sigma_\zeta}{\sigma_\zeta\kappa_\zeta + \kappa_\zeta^2\alpha_\zeta}\Big(c_\zeta - 1\Big). \tag{5}$$

The conventional ADI-FDTD with CFS-CPML still involves matrix operators $\mathbf{A}'$ and $\mathbf{B}'$ on the right-hand-sides (c.f. (4a) and (4c)). To remove $\mathbf{A}'$ and $\mathbf{B}'$ from the right-hand-sides, we now formulate the

FADI-FDTD scheme as

$$\mathbf{v}^n = \tilde{\mathbf{u}}^n - \mathbf{v}^{n-\frac{1}{2}} + \frac{\Delta t}{2}\mathbf{W}\Psi^n \tag{6a}$$

$$\left(\frac{1}{2}\mathbf{I} - \frac{\Delta t}{4}\left(\mathbf{A}' + \frac{\mathbf{L}}{2}\right)\right)\tilde{\mathbf{u}}^{n+\frac{1}{2}} = \mathbf{v}^n \tag{6b}$$

$$\Psi^{n+\frac{1}{2}} = \mathbf{C}\Psi^n + \frac{\mathbf{D}}{2}\tilde{\mathbf{u}}^{n+\frac{1}{2}} \tag{6c}$$

$$\mathbf{v}^{n+\frac{1}{2}} = \tilde{\mathbf{u}}^{n+\frac{1}{2}} - \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{W}\Psi^{n+\frac{1}{2}} \tag{6d}$$

$$\left(\frac{1}{2}\mathbf{I} - \frac{\Delta t}{4}\left(\mathbf{B}' + \frac{\mathbf{L}}{2}\right)\right)\tilde{\mathbf{u}}^{n+1} = \mathbf{v}^{n+\frac{1}{2}} \tag{6e}$$

$$\Psi^{n+1} = \mathbf{C}\Psi^{n+\frac{1}{2}} + \frac{\mathbf{D}}{2}\tilde{\mathbf{u}}^{n+1} \tag{6f}$$

where

$$\tilde{\mathbf{u}} = \begin{bmatrix}\tilde{E}_x & \tilde{E}_y & \tilde{E}_z & \tilde{H}_x & \tilde{H}_y & \tilde{H}_z\end{bmatrix}^T,$$
$$\mathbf{v} = \begin{bmatrix}e_x & e_y & e_z & h_x & h_y & h_z\end{bmatrix}^T$$

with initialization $\mathbf{v}^{-\frac{1}{2}} = (\mathbf{I} - \frac{\Delta t}{2}(\mathbf{B}' + \frac{\mathbf{L}}{2}))\mathbf{u}^0$ in the main grid. Note that $\tilde{\mathbf{u}} = 2\mathbf{u}$ and $\mathbf{v}$'s serve as temporary auxiliary field variables which do not require additional memory [8]. It can be seen now that the algorithm has its right-hand-sides free of matrix operators $\mathbf{A}'$ and $\mathbf{B}'$. This results in reduction of the number of update coefficients and field variables (shown later).

Assuming $\sigma^* = 0$, the update equations for e_x, h_z, $\tilde{E}_x$, $\tilde{H}_z$ and $\psi_{e_{xy}}$ for the first sub-step (6a)–(6c) are as follows (other field equations can be written down by permuting the indices)

$$e^n_{x_{i+\frac{1}{2},j,k}} = \tilde{E}^n_{x_{i+\frac{1}{2},j,k}} - e^{n-\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}} + a_1\left(\psi^n_{e_{xy}{}_{i+\frac{1}{2},j,k}} - \psi^n_{e_{xz}{}_{i+\frac{1}{2},j,k}}\right) \tag{7a}$$

$$\begin{aligned} h^n_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} &= \tilde{H}^n_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} - h^{n-\frac{1}{2}}_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} \\ &\quad + a_2\left(\psi^n_{h_{zy}{}_{i+\frac{1}{2},j+\frac{1}{2},k}} - \psi^n_{h_{zx}{}_{i+\frac{1}{2},j+\frac{1}{2},k}}\right) \end{aligned} \tag{7b}$$

$$\tilde{E}^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}} = \frac{2}{\beta}e^n_{x_{i+\frac{1}{2},j,k}} + \frac{a_{1,y}}{\kappa_{y_j}\beta}\left(\tilde{H}^{n+\frac{1}{2}}_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} - \tilde{H}^{n+\frac{1}{2}}_{z_{i+\frac{1}{2},j-\frac{1}{2},k}}\right) \tag{7c}$$

$$\tilde{H}^{n+\frac{1}{2}}_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} = 2h^n_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} + \frac{a_{2,y}}{\kappa_{y_{j+\frac{1}{2}}}}\left(\tilde{E}^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j+1,k}} - \tilde{E}^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}}\right) \tag{7d}$$

$$\psi^{n+\frac{1}{2}}_{e_{xy}{}_{i+\frac{1}{2},j,k}} = c_{y_j}\psi^n_{e_{xy}{}_{i+\frac{1}{2},j,k}} + \frac{d_{y_j}}{2\Delta y}\left(\tilde{H}^{n+\frac{1}{2}}_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} - \tilde{H}^{n+\frac{1}{2}}_{z_{i+\frac{1}{2},j-\frac{1}{2},k}}\right) \tag{7e}$$

where

$$a_1 = \frac{\Delta t}{2\epsilon}, \quad a_2 = \frac{\Delta t}{2\mu}, \quad a_{1,\zeta} = \frac{a_1}{\Delta\zeta}, \quad a_{2,\zeta} = \frac{a_2}{\Delta\zeta}, \quad \beta = \Big(1 + a_1\frac{\sigma}{2}\Big),$$

and $\Delta\zeta$ is the cell size. Note that for simplicity, we have omitted the subscript indices for media parameters ϵ, μ and σ.

By substituting (7d) into (7c), we arrive at the implicit update equation of E_x as

$$\begin{aligned}
&\frac{-a_{1,y}a_{2,y}}{2\kappa_{y_j}\kappa_{y_{j-\frac{1}{2}}}\beta}\tilde{E}^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j-1,k}} - \frac{a_{1,y}a_{2,y}}{2\kappa_{y_j}\kappa_{y_{j+\frac{1}{2}}}\beta}\tilde{E}^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j+1,k}} + \frac{\gamma_{y_j}}{2}\tilde{E}^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}} \\
&= \frac{1}{\beta}e^{n}_{x_{i+\frac{1}{2},j,k}} + \frac{a_{1,y}}{\kappa_{y_j}\beta}\Big(h^{n}_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} - h^{n}_{z_{i+\frac{1}{2},j-\frac{1}{2},k}}\Big) \qquad (8)
\end{aligned}$$

with

$$\gamma_{y_j} = 1 + \frac{a_{1,y}a_{2,y}}{\kappa_{y_j}\kappa_{y_{j+\frac{1}{2}}}\beta} + \frac{a_{1,y}a_{2,y}}{\kappa_{y_j}\kappa_{y_{j-\frac{1}{2}}}\beta}.$$

For the second sub-step, the update equations for e_x, h_y, $\tilde{E}_x$, $\tilde{H}_y$ and $\psi_{e_{xz}}$ are as follows (other field equations can be written down by permuting the indices):

$$e^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}} = \tilde{E}^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}} - e^{n}_{x_{i+\frac{1}{2},j,k}} + a_1\Big(\psi^{n+\frac{1}{2}}_{e_{xy}{}_{i+\frac{1}{2},j,k}} - \psi^{n+\frac{1}{2}}_{e_{xz}{}_{i+\frac{1}{2},j,k}}\Big) \qquad (9a)$$

$$\begin{aligned}
h^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} &= \tilde{H}^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} - h^{n}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} \\
&\quad + a_2\Big(\psi^{n+\frac{1}{2}}_{h_{yx}{}_{i+\frac{1}{2},j,k+\frac{1}{2}}} - \psi^{n+\frac{1}{2}}_{h_{yz}{}_{i+\frac{1}{2},j,k+\frac{1}{2}}}\Big) \qquad (9b)
\end{aligned}$$

$$\tilde{E}^{n+1}_{x_{i+\frac{1}{2},j,k}} = \frac{2}{\beta}e^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}} - \frac{a_{1,z}}{\kappa_{z_k}\beta}\Big(\tilde{H}^{n+1}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} - \tilde{H}^{n+1}_{y_{i+\frac{1}{2},j,k-\frac{1}{2}}}\Big) \qquad (9c)$$

$$\tilde{H}^{n+1}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} = 2h^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} - \frac{a_{2,z}}{\kappa_{z_{k+\frac{1}{2}}}}\Big(\tilde{E}^{n+1}_{x_{i+\frac{1}{2},j,k+1}} - \tilde{E}^{n+1}_{x_{i+\frac{1}{2},j,k}}\Big) \qquad (9d)$$

$$\psi^{n+1}_{e_{xz}{}_{i+\frac{1}{2},j,k}} = c_{z_k}\psi^{n+\frac{1}{2}}_{e_{xz}{}_{i+\frac{1}{2},j,k}} + \frac{d_{z_k}}{2\Delta z}\Big(\tilde{H}^{n+1}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} - \tilde{H}^{n+1}_{y_{i+\frac{1}{2},j,k-\frac{1}{2}}}\Big) \qquad (9e)$$

Using similar approach, the implicit update equation for E_x is derived as

$$\begin{aligned}
&\frac{-a_{1,z}a_{2,z}}{2\kappa_{z_k}\kappa_{z_{k-\frac{1}{2}}}\beta}\tilde{E}^{n+1}_{x_{i+\frac{1}{2},j,k-1}} - \frac{a_{1,z}a_{2,z}}{2\kappa_{z_k}\kappa_{z_{k+\frac{1}{2}}}\beta}\tilde{E}^{n+1}_{x_{i+\frac{1}{2},j,k+1}} + \frac{\gamma_{z_k}}{2}\tilde{E}^{n+1}_{x_{i+\frac{1}{2},j,k}} \\
&= \frac{1}{\beta}e^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}} - \frac{a_{1,z}}{\kappa_{z_k}\beta}\Big(h^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} - h^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k-\frac{1}{2}}}\Big) \qquad (10)
\end{aligned}$$

with

$$\gamma_{z_k} = 1 + \frac{a_{1,z}a_{2,z}}{\kappa_{z_k}\kappa_{z_{k+\frac{1}{2}}}\beta} + \frac{a_{1,z}a_{2,z}}{\kappa_{z_k}\kappa_{z_{k-\frac{1}{2}}}\beta}.$$

For comparison, the update equations for the conventional ADI-FDTD with CFS-CPML is provided in Appendix A. We can see that in FADI-FDTD, ψ is only required in the auxilliary field update equations (c.f. (7a), (7b), (9a) and (9b)) and NOT in the implicit update equations of electric fields (c.f. (8) and (10)). Not only that, ψ_e and ψ_h are well separated in the FADI-FDTD update equations (i.e., ψ_e is only needed in electric field update equation while ψ_h is only needed in magnetic field update equation). On the other hand, for conventional ADI-FDTD, both ψ_e and ψ_h are needed simultaneously at the implicit electric field update equations as evident from (A4) and (A5). Furthermore, the number of overall right-hand-side terms in the conventional ADI-FDTD update equations are higher compared to that of FADI-FDTD, which results in more arithmetic and memory indexing operations. For all these advantages, our FADI-FDTD is very attractive for its better conciseness, efficiency and programming simplicity. To further increase the efficiency, we incorporate the FADI-FDTD into GPU using CUDA as will be shown in the subsequent sections.

3. GPU-ACCELERATED FADI-FDTD WITH CFS-CPML

3.1. CUDA Architecture

CUDA is a parallel computing architecture developed by NVIDIA [11–14], which is accessible to software developers through industry standard programming languages. A GPU with CUDA capabilities has a set of build-in streaming multiprocessors. Each streaming multiprocessor consists of a shared memory (which can be used to share data between the threads within a thread block), a set of 32-bit registers and read-only caches. Each multiprocessor contains 8 streaming processors, and every streaming processor supports parallel executing model of the single-instruction multiple-data (SIMD). The Random Access Memory (RAM) located on the GPU card serves as the global memory used to store the massive data.

3.2. CUDA Programming Model

A CUDA program is executed on both the host (CPU) and device (GPU). The set of instructions that exhibit rich amount of data parallelism are implemented in the device code, whereas the set of

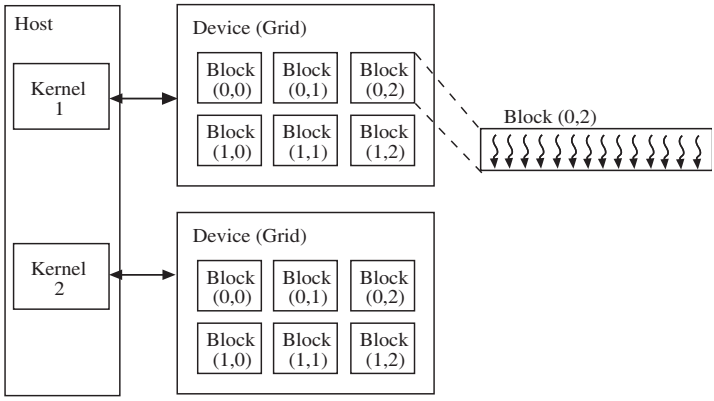

Figure 1. Execution of a typical CUDA program.

instructions with little or no data parallelism are implemented in host code.

The execution of a typical CUDA program (illustrated in Fig. 1) starts with host execution [14]. When a kernel function is invoked, the execution is moved to a device, where a large number of threads are generated to take advantage of abundant data parallelism. All the threads that are generated by a kernel are organized as a grid of thread blocks and each block consists of a maximum of 512 threads. The grid of thread blocks is then executed on the GPU by assigning blocks for execution on the streaming multiprocessors. When the execution of a kernel is completed, the corresponding grid terminates and the execution returns to the host until another kernel is invoked.

3.3. Parallelization for Gaussian Elimination Using LU Factorization

We make use of the 2-dimensional grid and 1-dimensional threads for the field vector E_x field along the x-, y-, and z-directions. In the first procedure of the FADI-FDTD update equations, $E_{x_{i+\frac{1}{2},j+1,k}}$ requires the updated value of $E_{x_{i+\frac{1}{2},j,k}}$ along the y-direction. This makes E_x along x- and z-directions possible for parallelism. Conversely, in the second procedure, $E_{x_{i+\frac{1}{2},j,k+1}}$ requires the updated value of $E_{x_{i+\frac{1}{2},j,k}}$ along the z-direction. This in turn makes E_x along x- and y-directions possible for parallelism. The parallelization of FADI-FDTD is realized through gaussian elimination using LU Factorization method [15]. In order to exploit the data parallelism in these directions, we have

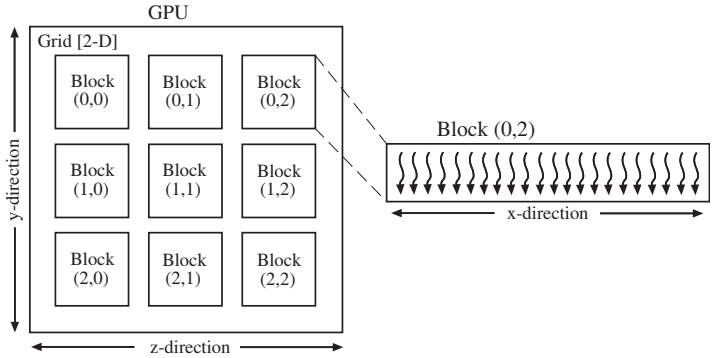

Figure 2. CUDA threads organization for E_x.

arranged the field vector for E_x in CUDA according to the arrangement shown in Fig. 2. For instance, in the first procedure, blocks from the same row are run in parallel (data parallelism in x- and z-directions). Starting from the first row, the process will move to the subsequent row after all blocks from the preceeding row are processed. This row by row procedure will continue until the maximum number of grids in y-direction is reached. On the other hand, in the second procedure, blocks from the same column are run in parallel (data parallelism in x- and y-directions). The process is repeated column by column until the maximum number of grids in z-direction is reached.

The update procedure of the GPU-accelerated FADI-FDTD with CFS-CPML using CUDA is depicted as a flowchart in Fig. 3. Note that je and ke are the number of cells in the y- and z-direction, respectively. For the update equations of the E fields, we only show those of E_x. The remaining procedures for E_y and E_z fields can be obtained by permutting the indices. As mentioned, the ψ's of FADI-FDTD are now included in the auxilliary update equations. Therefore, the main update equations of E fields are the same as the FADI-FDTD without CFS-CPML, which feature great simplicity and convenience.

4. NUMERICAL RESULTS

In this section, the performance of the CPU and GPU-accelerated FADI-FDTD with CFS-CPML in free space are illustrated through numerical simulations. The computation domain has a dimension of $42 \times 42 \times 42$ and cell size $\Delta x = \Delta y = \Delta z = 1.0\,\text{mm}$. Ten cells of PML are applied to all six sides of the lattice. Within the PML, the

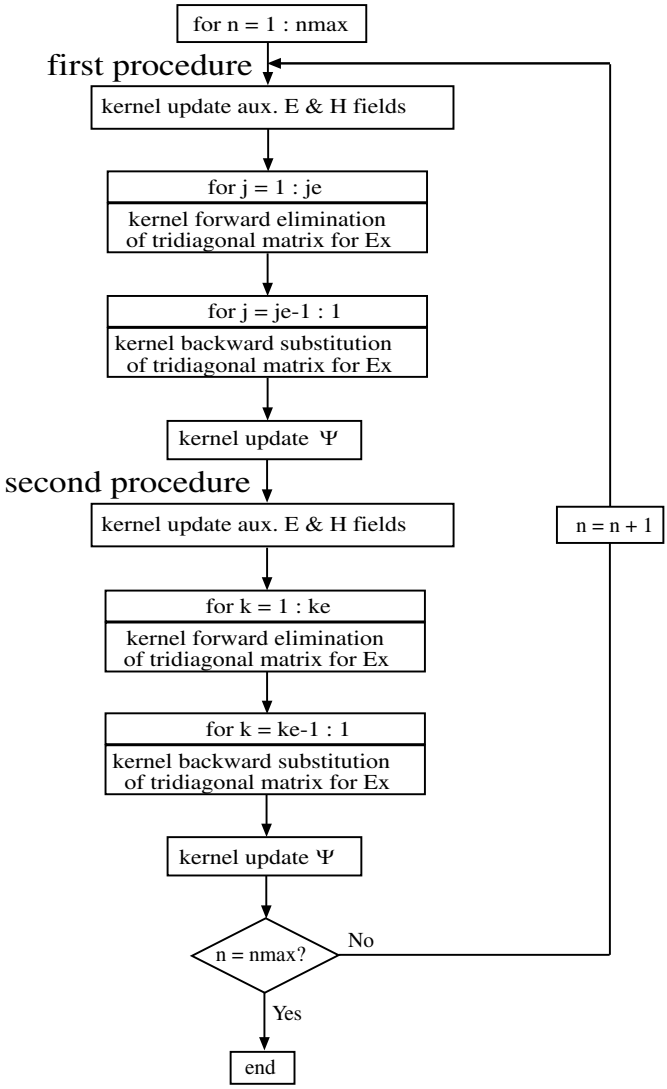

Figure 3. Flowchart of FADI-FDTD with CFS-CPML as implemented on GPU.

consitutive parameters were scaled using polynomial scaling

$$\sigma_\zeta(\zeta) = \frac{\sigma_{\zeta_{\max}} \mid \zeta - \zeta_0 \mid^m}{\delta^m} \tag{11}$$

$$\kappa_\zeta(\zeta) = 1 + (\kappa_{\zeta_{\max}} - 1)\frac{\mid \zeta - \zeta_0 \mid^m}{\delta^m} \tag{12}$$

where δ is the thickness of the PML absorber, ζ_0 is the interface to free space and m is the order of the polynomial.

A choice for $\sigma_{\zeta_{\max}}$ that will minimize reflection is expressed as [5]

$$\sigma_{\zeta_{\max}} = \sigma_{opt} \approx \frac{m+1}{150\pi\Delta\zeta}. \tag{13}$$

For our study, the PML parameters are $\sigma_{\zeta_{\max}} = \sigma_{opt} = 10.61\,\mathrm{S/m}$, $\kappa_{\zeta_{\max}} = 15$, $\alpha_\zeta = 0.08\,\mathrm{S/m}$, and $m = 4$.

A small dipole source excitation is located at the center of the computation region. The source is a modulated Gaussian pulse applied to the E_z component as

$$J_z = e^{-(\frac{t-t_0}{\tau})^2} \sin\left(2\pi f_c(t-t_0)\right), \tag{14}$$

where $\tau = 160\,\mathrm{ps}$, $t_0 = 3\tau$, $f_c = 3.175\,\mathrm{GHz}$.

The reflection error for PML is studied with the observation point located ten cells away from the source and one cell away from the PML interface. The reflection error for the PML is evaluated by

$$R = 20\log_{10}\left(\frac{|E - E_{ref}|}{\max|E_{ref}|}\right) \tag{15}$$

where E_{ref} is the reference value calculated in a grid large enough so that any reflection from the boundary is isolated.

For comparison, Fig. 4 presents the electric field's time domain characteristic at the observation point for both CPU and GPU-accelerated FADI-FDTD with CFS-CPML. CFLN is chosen as 4, where $CFLN = \Delta t/\Delta t_{CFL}$ and $\Delta t_{CFL} = 1.926\,\mathrm{ps}$ is the CFL limit in Yee's explicit FDTD method. The good agreement indicates that the GPU-accelerated FADI-FDTD is correct although the GPU adopts single precision floating-point format as compared to the double-precision floating-point format of the CPU.

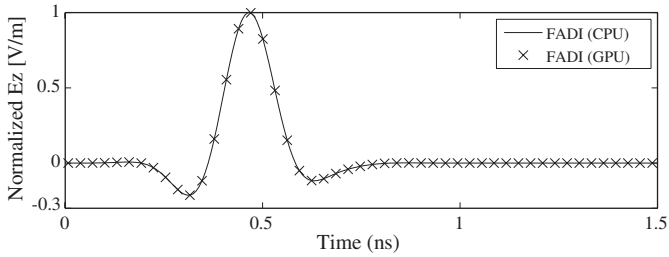

Figure 4. Comparison of time domain waveforms for electric field at observation point (CFLN = 4).

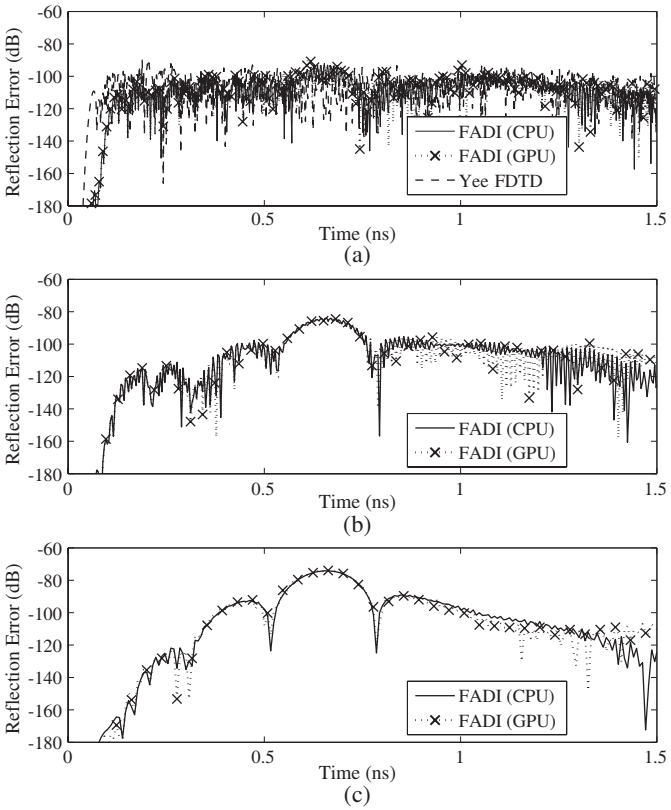

Figure 5. Reflection errors of CPU and GPU-accelerated FADI-FDTD with CFS-CPML for (a) CFLN = 1, (b) CFLN = 2 and (c) CFLN = 4.

Figure 5 illustrates the reflection errors of CPU and GPU-accelerated FADI-FDTD with CFS-CPML for various CFLNs. The maximum amplitude of $|E_{ref}|$ used in (15) for CFLN 1, 2 and 4 are given as 0.0655, 0.0691 and 0.0857 V/m, respectively. For CFLN = 1, the FADI- and Yee-FDTD with CFS-CPML exhibit reflection errors of the same order. We also observe that the reflection errors computed using both CPU and GPU implementations agree well with each other, and the maximum reflection error is around -75 dB for CFLN = 4. Such low reflection error shows the effectiveness of the CFS-CPML for absorbing outgoing electromagnetic wave. Both CPU and GPU implementations are equally effective, with greater efficiency for the latter. The slight discrepancies of reflection errors are the result of single-precision floating-point format currently supported by the low cost GPU.

Table 1. Comparison of CPU and GPU time for FADI-FDTD and Yee-FDTD with CFS-CPML (For FADI-FDTD, CFLN = 4, 250 iterations. For Yee-FDTD, CFLN = 1, 1000 iterations)

Scheme	Domain Size	CPU Time (s)	GPU Time (s)	Speedup
FADI-FDTD with CFS-CPML	100^3	679.53	60.74	11.19
	150^3	2842.11	201.31	14.12
	200^3	6828.12	463.88	14.72
Yee-FDTD with CFS-CPML	100^3	1011.70	79.92	12.66
	150^3	4810.23	263.27	18.27
	200^3	11504.21	621.39	18.51

We further compare the CPU and GPU time of the FADI-FDTD with CFS-CPML, c.f. Table 1. The numerical simulation is performed for 250 iterations at CFLN = 4 in various computation domains. Note that 250 time iterations at CFLN = 4 is more than sufficient for the entire wave motion to reach the PML layer. The computing platform is C++ environment with Intel Dual Core 2.66 GHz processor and a NVIDIA GeForce GTS250 GPU. From Table 1, we find that the efficiency gain achieved by the GPU over CPU is approximately $11 \sim 15$ times. It is thus ascertained that substantial gain is achievable for GPU by exploiting data parallelism. The numerical results for Yee-FDTD with CFS-CPML are also included in Table 1, for 1000 iterations at CFLN = 1. The efficiency gain achieved by the GPU over CPU of Yee-FDTD is slightly higher than that of FADI-FDTD. This is because Yee-FDTD is a fully explicit scheme which allow data parallelism in three directions (x, y and z) while for FADI-FDTD, the data parallelism is in two directions as discussed in Section 3.3. Nevertheless, the overall CPU and GPU time incurred by Yee-FDTD is still higher than that of FADI-FDTD, due to its restricted time step caused by the CFL stability condition.

5. CONCLUSION

This paper has presented the GPU-accelerated FADI-FDTD with CFS-CPML. The compact matrix form of the conventional ADI-FDTD method with CFS-CPML has been formulated into FADI-FDTD with its right-hand-sides free of matrix operators, resulting in simpler and more concise update equations. Using CUDA, the FADI-FDTD with

CFS-CPML has been further incorporated into the GPU to exploit data parallelism. Numerical results have shown that a much higher efficiency gain of up to 15 times can be achieved.

Like FADI-FDTD, the locally one-dimensional (LOD) FDTD with CFS-CPML could also be formulated in the most fundamental form with concise matrix-operator-free right-hand-sides. However, more work is required to simplify the equations further and this shall be addressed in future work.

APPENDIX A. (UPDATE EQUATIONS FOR CONVENTIONAL ADI-FDTD)

Based on (4a)–(4b), the update equations for conventional ADI-FDTD from n to $n+\frac{1}{2}$ are written as (other field equations can be written down by permuting the indices)

$$\begin{aligned} E_{x_{i+\frac{1}{2},j,k}}^{n+\frac{1}{2}} &= \frac{\chi}{\beta} E_{x_{i+\frac{1}{2},j,k}}^{n} - \frac{a_{1,z}}{\kappa_{z_k}\beta}\left(H_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}}^{n} - H_{y_{i+\frac{1}{2},j,k-\frac{1}{2}}}^{n}\right) \\ &\quad + \frac{a_{1,y}}{\kappa_{y_j}\beta}\left(H_{z_{i+\frac{1}{2},j+\frac{1}{2},k}}^{n+\frac{1}{2}} - H_{z_{i+\frac{1}{2},j-\frac{1}{2},k}}^{n+\frac{1}{2}}\right) \\ &\quad + \frac{a_1}{\beta}\left(\psi_{e_{xy}{}_{i+\frac{1}{2},j,k}}^{n} - \psi_{e_{xz}{}_{i+\frac{1}{2},j,k}}^{n}\right) \end{aligned} \tag{A1}$$

$$\begin{aligned} H_{z_{i+\frac{1}{2},j+\frac{1}{2},k}}^{n+\frac{1}{2}} &= H_{z_{i+\frac{1}{2},j+\frac{1}{2},k}}^{n} - \frac{a_{2,x}}{\kappa_{x_{i+\frac{1}{2}}}}\left(E_{y_{i+1,j+\frac{1}{2},k}}^{n} - E_{y_{i,j+\frac{1}{2},k}}^{n}\right) \\ &\quad + \frac{a_{2,y}}{\kappa_{y_{j+\frac{1}{2}}}}\left(E_{x_{i+\frac{1}{2},j+1,k}}^{n+\frac{1}{2}} - E_{x_{i+\frac{1}{2},j,k}}^{n+\frac{1}{2}}\right) \\ &\quad + a_2\left(\psi_{h_{zy}{}_{i+\frac{1}{2},j+\frac{1}{2},k}}^{n} - \psi_{h_{zx}{}_{i+\frac{1}{2},j+\frac{1}{2},k}}^{n}\right) \end{aligned} \tag{A2}$$

$$\psi_{e_{xy}{}_{i+\frac{1}{2},j,k}}^{n+\frac{1}{2}} = c_{y_j}\psi_{e_{xy}{}_{i+\frac{1}{2},j,k}}^{n} + \frac{d_{y_j}}{\Delta y}\left(H_{z_{i+\frac{1}{2},j+\frac{1}{2},k}}^{n+\frac{1}{2}} - H_{z_{i+\frac{1}{2},j-\frac{1}{2},k}}^{n+\frac{1}{2}}\right) \tag{A3}$$

where $\chi = \left(1 - a_1\frac{\sigma}{2}\right)$.

By substituting (A2) into (A1), the implicit update equation of

E_x can be obtained as

$$\begin{aligned}
&\frac{-a_{1,y}a_{2,y}}{\kappa_{y_j}\kappa_{y_{j-\frac{1}{2}}}\beta}E^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j-1,k}} - \frac{a_{1,y}a_{2,y}}{\kappa_{y_j}\kappa_{y_{j+\frac{1}{2}}}\beta}E^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j+1,k}} + \gamma_{y_j}E^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}} \\
&= \frac{\chi}{\beta}E^{n}_{x_{i+\frac{1}{2},j,k}} - \frac{a_{1,z}}{\kappa_{z_k}\beta}\left(H^{n}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} - H^{n}_{y_{i+\frac{1}{2},j,k-\frac{1}{2}}}\right) \\
&-\frac{a_{1,y}a_{2,x}}{\kappa_{y_j}\kappa_{x_{i+\frac{1}{2}}}\beta}\left(E^{n}_{y_{i+1,j+\frac{1}{2},k}} - E^{n}_{y_{i,j+\frac{1}{2},k}} - E^{n}_{y_{i+1,j-\frac{1}{2},k}} + E^{n}_{y_{i,j-\frac{1}{2},k}}\right) \\
&+\frac{a_{1,y}}{\kappa_{y_j}\beta}\left(H^{n}_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} - H^{n}_{z_{i+\frac{1}{2},j-\frac{1}{2},k}}\right) + \frac{a_1}{\beta}\left(\psi^{n}_{e_{xy}{}_{i+\frac{1}{2},j,k}} - \psi^{n}_{e_{xz}{}_{i+\frac{1}{2},j,k}}\right) \\
&+\frac{a_{1,y}a_2}{\kappa_{y_j}\beta}\left(\psi^{n}_{h_{zy}{}_{i+\frac{1}{2},j+\frac{1}{2},k}} - \psi^{n}_{h_{zx}{}_{i+\frac{1}{2},j+\frac{1}{2},k}} - \psi^{n}_{h_{zy}{}_{i+\frac{1}{2},j-\frac{1}{2},k}} + \psi^{n}_{h_{zx}{}_{i+\frac{1}{2},j-\frac{1}{2},k}}\right) \quad \text{(A4)}
\end{aligned}$$

For the second procedure, the implicit update equation of E_x can be derived from (4c) as (other field equations can be written down by permuting the indices)

$$\begin{aligned}
&\frac{-a_{1,z}a_{2,z}}{\kappa_{z_k}\kappa_{z_{k-\frac{1}{2}}}\beta}E^{n+1}_{x_{i+\frac{1}{2},j,k-1}} - \frac{a_{1,z}a_{2,z}}{\kappa_{z_k}\kappa_{z_{k+\frac{1}{2}}}\beta}E^{n+1}_{x_{i+\frac{1}{2},j,k+1}} + \gamma_{z_k}E^{n+1}_{x_{i+\frac{1}{2},j,k}} \\
&= \frac{\chi}{\beta}E^{n+\frac{1}{2}}_{x_{i+\frac{1}{2},j,k}} + \frac{a_{1,y}}{\kappa_{y_j}\beta}\left(H^{n+\frac{1}{2}}_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} - H^{n+\frac{1}{2}}_{z_{i+\frac{1}{2},j-\frac{1}{2},k}}\right) \\
&-\frac{a_{1,z}a_{2,x}}{\kappa_{z_k}\kappa_{x_{i+\frac{1}{2}}}\beta}\left(E^{n+\frac{1}{2}}_{z_{i+1,j,k+\frac{1}{2}}} - E^{n+\frac{1}{2}}_{z_{i,j,k+\frac{1}{2}}} - E^{n+\frac{1}{2}}_{z_{i+1,j,k-\frac{1}{2}}} + E^{n+\frac{1}{2}}_{z_{i,j,k-\frac{1}{2}}}\right) \\
&-\frac{a_{1,z}}{\kappa_{z_k}\beta}\left(H^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} - H^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k-\frac{1}{2}}}\right) + \frac{a_1}{\beta}\left(\psi^{n+\frac{1}{2}}_{e_{xy}{}_{i+\frac{1}{2},j,k}} - \psi^{n+\frac{1}{2}}_{e_{xz}{}_{i+\frac{1}{2},j,k}}\right) \\
&-\frac{a_{1,z}a_2}{\kappa_{z_k}\beta}\left(\psi^{n+\frac{1}{2}}_{h_{yx}{}_{i+\frac{1}{2},j,k+\frac{1}{2}}} - \psi^{n+\frac{1}{2}}_{h_{yz}{}_{i+\frac{1}{2},j,k+\frac{1}{2}}} - \psi^{n+\frac{1}{2}}_{h_{yx}{}_{i+\frac{1}{2},j,k-\frac{1}{2}}} + \psi^{n+\frac{1}{2}}_{h_{yz}{}_{i+\frac{1}{2},j,k-\frac{1}{2}}}\right) \quad \text{(A5)}
\end{aligned}$$

REFERENCES

1. Zheng, F., Z. Chen, and J. Zhang, "Toward the development of a three-dimensional unconditionally stable finite-difference time-domain method," *IEEE Trans. Microwave Theory Tech.*, Vol. 48, No. 9, 1550–1558, Sep. 2000.
2. Namiki, T., "3-D ADI-FDTD method — Unconditionally stable time-domain algorithm for solving full vector Maxwell's equations," *IEEE Trans. Microwave Theory Tech.*, Vol. 48, No. 10, 1743–1748, Oct. 2000.
3. Tay, W. C. and E. L. Tan, "Implementation of the Mur first order absorbing boundary condition in efficient 3-D ADI-FDTD," *IEEE Antennas and Propag. Society Int. Symp.*, 1–4, Jun. 2009.

4. Tay, W. C. and E. L. Tan, "Mur absorbing boundary condition for efficient fundamental 3-D LOD-FDTD," *IEEE Microw. Wireless Comp. Lett.*, Vol. 20, No. 2, 61–63, Feb. 2010.
5. Berenger, J. P., "A perfectly matched layer for the absorption of electromagnetic waves," *J. Comput. Phys.*, Vol. 114, 185–200, Oct. 1994.
6. Gedney, S. D., G. Liu, J. Alan Rodden, and A. Zhu, "Perfectly matched layer media with CFS for an unconditionally stable ADI-FDTD method," *IEEE Trans. Antennas Propagat.*, Vol. 49, No. 11, 1554–1559, Nov. 2001.
7. Tan, E. L., "Efficient algorithm for the unconditionally stable 3-D ADI-FDTD method," *IEEE Microw. Wireless Comp. Lett.*, Vol. 17, No. 1, 7–9, Jan. 2007.
8. Tan, E. L., "Fundamental schemes for efficient unconditionally stable implicit finite-difference time-domain methods," *IEEE Trans. Antennas Propagat.*, Vol. 56, No. 1, 170–177, Jan. 2008.
9. Inman, M. J. and A. Z. Elsherbeni, "Programming video cards for computational electromagnetics applications," *Antennas Propag. Mag.*, Vol. 47, 71–78, Dec. 2005.
10. Tao, Y.-B., H. Lin, and H. J. Lin, "From CPU to GPU: GPU-based electromagnetic computing (GPUECO)," *Progress In Electromagnetics Research*, Vol. 81, 1–19, 2008.
11. Pharr, M. and R. Fernando, *GPUGems2: Programming Techniques for High-performance Graphics and General-purpose Computation*, Addison-Wesley, 2005.
12. Owens, J. D., M. Houston, et al., "GPU computing," *Proceedings of the IEEE*, Vol. 96, No. 5, 879–899, May 2008.
13. Nvidia Corporation, *NVIDIA CUDA Programming Guide*, Version 2.3, Aug. 2009.
14. Kirk, D. B. and W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, Morgan Kaufmann, 2010.
15. Isaacson, E., *Analysis of Numerical Methods*, Dover Publication, 1994.