Published in IEEE Transactions Antennas and propagation,
vol. 62(3), pp. 1330-1338, 2014.

Efficient Parallel LOD-FDTD Method for Debye-Dispersive Media

Tadashi Hemmi, Fumie Costen, *Senior Member, IEEE*, Salvador Garcia, *Member, IEEE*, Ryutaro Himeno, Hideo Yokota, and Mehshan Mustafa

***Abstract*—The locally one-dimensional finite-difference time-domain (LOD-FDTD) method is a promising implicit technique for solving Maxwell's equations in numerical electromagnetics. This paper describes an efficient message passing interface (MPI)-parallel implementation of the LOD-FDTD method for Debye-dispersive media. Its computational efficiency is demonstrated to be superior to that of the parallel ADI-FDTD method. We demonstrate the effectiveness of the proposed parallel algorithm in the simulation of a bio-electromagnetic problem: the deep brain stimulation (DBS) in the human body.**

***Index Terms*—Distributed memory systems, electromagnetic fields, electromagnetic propagation in dispersive media, finite-difference methods, numerical analysis, parallel programming, time-domain analysis.**

I. Introduction

COMPUTATIONAL electromagnetic simulators have become an invaluable tool with applications ranging from telecommunications to radar systems and design of high-speed electronic circuit boards as well as healthcare device in biomedical engineering. There exist several approaches to solve Maxwell equations numerically. Among them, the finite-difference time-domain (FDTD) method has become the most widely used [1].

Even though the FDTD method is flexible and robust and easy to parallelize, its computational efficiency is limited by the Courant–Friedrich–Levy (CFL) stability condition [1]. This criterion imposes an upper limit on the maximum time-step Δt_{CFL}, depending on the minimum space-step, which may lead to large numbers of FDTD iterations. We face such a situation, for instance, in highly resonant problems or in complex problems requiring very fine spatial discretization.

There are two major approaches to improve the computational efficiency of the FDTD method, developing either hardware or algorithmic methodology. The hardware approach involves the parallelization of the computation by 1) using multiple cores in shared and/or distributed memory architectures (or graphics processing units [2]) and 2) taking advantage of modern processors' features such as the register level parallelization, i.e., the streaming SIMD extensions (SSEs) [3] or the advanced vector extensions (AVXs).

Following the algorithmic approach, the development of implicit formulations of the FDTD method, overcoming the CFL limit, has attracted great attention in the recent literature. Most of them are based on some variation of the Crank–Nicolson fully implicit FDTD (CN-FDTD) method [4], [5]. The most usual ones are the alternating direction implicit FDTD (ADI-FDTD) method [6], [7], and the locally one-dimensional FDTD (LOD-FDTD) method [8]–[15]. Implicit CN-FDTD-based methods are not constrained by the CFL stability condition and permit time-steps Δt only constrained by an accuracy criterion [16], over the CFL limit (i.e., $N_{\mathrm{CFL}} \triangleq (\Delta t/\Delta t_{\mathrm{CFL}}) > 1$), with potential computational gains. When they are combined with hardware acceleration techniques, they provide an efficient alternative to the classical FDTD method.

However, unlike parallel implementations of the FDTD method [17], CN-FDTD-based algorithms require data, which are not just one cell neighbors as is used for the explicit FDTD method, at each time-step. Thus, parallel implementation of these methods using techniques such as message passing interface (MPI) [18] results in a huge amount of data communication, hindering the scalability of the implementation.

There are efficient implementations of the parallel ADI-FDTD method [19], [20]. Nevertheless, their efficiency is limited by the fact that these methods require the alternative use of data along two-space directions at each time-step. On the other hand, the LOD-FDTD method only requires data along one direction, making it more attractive for parallelization.

In this paper, we present a new parallel MPI algorithm for the LOD-FDTD method, including Debye-dispersive media, and demonstrate its efficacy with a biomedical problem: the simulation of a deep brain stimulation (DBS) scenario, where the choice of precise positions of the transmitters and the stimulation waveforms, which are the key for the successful noninvasive stimulation, may be properly tuned with numerical simulations.

Manuscript received February 15, 2013; revised August 15, 2013; accepted August 29, 2013. Date of publication December 12, 2013; date of current version February 27, 2014. The work described in this paper and the research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013, under grant agreement no 205294 (HIRF SE project), and from the Spanish National Projects TEC2010-20841-C04-04, CSD2008-00068, and the Junta de Andalucia Project P09-TIC-5327.

T. Hemmi and M. Mustafa are with the School of Electrical and Electronic Engineering, University of Manchester, Manchester M13 9PL, U.K.

F. Costen with the School of Electrical and Electronic Engineering, University of Manchester, Manchester M13 9PL, U.K., and also with the Image Processing Research Team, Center for Advanced Photonics, RIKEN, Saitama 351-0198, Japan (e-mail: fumie.costen@manchester.ac.uk).

S. Garcia is with Faculty of Sciences, University of Granada, Granada E-18071, Spain.

R. Himeno is with the Advanced Center for Computing and Communication, RIKEN, Saitama 351-0198, Japan.

H. Yokota is with the Image Processing Research Team, Center for Advanced Photonics, RIKEN, Saitama 351-0198, Japan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAP.2013.2294860

The remainder of this paper is organized as follows. Section II shows the mathematical procedure for including Debye media in the LOD-FDTD method. Section III presents the approach to parallelize the LOD-FDTD method with Debye media on distributed memory architectures. The results of performance and scalability tests are given in Section IV and are compared with the parallel ADI-FDTD and classical FDTD methods. Section V demonstrates the applicability of the parallel LOD-FDTD method to a real problem, more specifically, the DBS problem mentioned previously. Section VI provides concluding remarks.

II. LOD-FDTD Method for Debye Media

The time-dependent Maxwell curl equations can be expressed for Debye media in a material independent form as

$$\nabla \times \boldsymbol{H} = \frac{\partial \boldsymbol{D}}{\partial t} \tag{1}$$

and

$$\nabla \times \boldsymbol{E} = \mu \frac{\partial \boldsymbol{H}}{\partial t} \tag{2}$$

together with the convolute constitutive relationship

$$\boldsymbol{D} = \epsilon \otimes \boldsymbol{E} \tag{3}$$

where $\boldsymbol{E}$, $\boldsymbol{H}$, and $\boldsymbol{D}$ are the electric and magnetic fields and the electric flux density; ϵ and μ are dispersive permittivity and permeability, respectively. The symbol $\otimes$ signifies convolution.

The permittivity of a one-pole Debye medium in the frequency-domain can be expressed as

$$\epsilon = \epsilon_0 \left(\epsilon_\infty + \frac{\epsilon_S - \epsilon_\infty}{1 + \jmath\omega\tau_D} + \frac{\sigma}{\jmath\omega\epsilon_0} \right) \tag{4}$$

where ϵ_∞ is the optical permittivity, ϵ_S is the static permittivity, τ_D is the characteristic relaxation time, σ is the static conductivity, and ω is the angular frequency.

The convolute relationship (3) can also be expressed in an auxiliary differential equation (ADE) form [21] as

$$\frac{\partial^2(\tau_D \boldsymbol{D})}{\partial t^2} + \frac{\partial \boldsymbol{D}}{\partial t} = \frac{\partial^2(\epsilon_0\epsilon_\infty\tau_D \boldsymbol{E})}{\partial t^2} + \frac{\partial(\epsilon_0\epsilon_S + \sigma\tau_D)\boldsymbol{E}}{\partial t} + \sigma \boldsymbol{E}. \tag{5}$$

As in the classical CN-FDTD algorithm, the LOD-FDTD method is developed by first using central differences for the time and space derivatives and averaging the fields affected by the curl operators in time. For instance, for the x component of (2) we find (similarly for the y- and the z-components of (2))

$$\begin{aligned} &\frac{1}{2}\left\{ \frac{E_z^{n+1}(i,j+1,k) - E_z^{n+1}(i,j,k)}{\Delta y} - \frac{E_y^{n+1}(i,j,k+1) - E_y^{n+1}(i,j,k)}{\Delta z} \right. \\ &\qquad \left. + \frac{E_z^{n}(i,j+1,k) - E_z^{n}(i,j,k)}{\Delta y} - \frac{E_y^{n}(i,j,k+1) - E_y^{n}(i,j,k)}{\Delta z} \right\} \\ &= -\mu \frac{H_x^{n+1}(i,j,k) - H_x^{n}(i,j,k)}{\Delta t} \end{aligned} \tag{6}$$

where Δx, Δy, and Δz are the spatial discretization in the x-, y-, and z-directions, respectively. The basis of the LOD-FDTD procedure consists of splitting the curl operator into each space direction and building a split-step time-marching algorithm [22]. For example, (6) advances in the y-direction using

$$\begin{aligned} &\frac{1}{2}\left\{ \frac{E_z^{n+\frac{2}{3}}(i,j+1,k) - E_z^{n+\frac{2}{3}}(i,j,k)}{\Delta y} + \frac{E_z^{n+\frac{1}{3}}(i,j+1,k) - E_z^{n+\frac{1}{3}}(i,j,k)}{\Delta y} \right\} \\ &= -\mu \frac{H_x^{n+\frac{2}{3}}(i,j,k) - H_x^{n+\frac{1}{3}}(i,j,k)}{\Delta t} \\ &\therefore H_x^{n+\frac{2}{3}}(i,j,k) \\ &= H_x^{n+\frac{1}{3}}(i,j,k) - \frac{\Delta t}{2\mu\Delta y} \\ &\quad \times \left\{ E_z^{n+\frac{2}{3}}(i,j+1,k) - E_z^{n+\frac{2}{3}}(i,j,k) + E_z^{n+\frac{1}{3}}(i,j+1,k) - E_z^{n+\frac{1}{3}}(i,j,k) \right\} \end{aligned} \tag{7}$$

and in the z-direction as

$$\begin{aligned} &\frac{1}{2}\left\{ -\frac{E_y^{n+1}(i,j,k+1) - E_y^{n+1}(i,j,k)}{\Delta z} - \frac{E_y^{n+\frac{2}{3}}(i,j,k+1) - E_y^{n+\frac{2}{3}}(i,j,k)}{\Delta z} \right\} \\ &= -\mu \frac{H_x^{n+1}(i,j,k) - H_x^{n+\frac{2}{3}}(i,j,k)}{\Delta t} \\ &\therefore H_x^{n+1}(i,j,k) \\ &= H_x^{n+\frac{2}{3}}(i,j,k) + \frac{\Delta t}{2\mu\Delta z} \\ &\quad \times \left\{ E_y^{n+1}(i,j,k+1) - E_y^{n+1}(i,j,k) + E_y^{n+\frac{2}{3}}(i,j,k+1) - E_y^{n+\frac{2}{3}}(i,j,k) \right\}. \end{aligned} \tag{8}$$

In the same way, spatial and temporal discretization is applied to (1) and (5), and the term of $\sigma\boldsymbol{E}$ in (5) is averaged over time. After the discretization, the z-component of (1) is expressed as

$$\begin{aligned} &\frac{1}{2}\left\{ \frac{H_y^{n+1}(i,j,k) - H_y^{n+1}(i-1,j,k)}{\Delta x} - \frac{H_x^{n+1}(i,j,k) - H_x^{n+1}(i,j-1,k)}{\Delta y} \right. \\ &\qquad \left. + \frac{H_y^{n}(i,j,k) - H_y^{n}(i-1,j,k)}{\Delta x} - \frac{H_x^{n}(i,j,k) - H_x^{n}(i,j-1,k)}{\Delta y} \right\} \\ &= \frac{D_z^{n+1}(i,j,k) - D_z^{n}(i,j,k)}{\Delta t} \end{aligned} \tag{9}$$

and the z component of (5) is described as

$$\begin{aligned}\tau_{\mathrm{D}(i,j,k)} & \frac{D_z^{n+1}(i,j,k) - 2D_z^n(i,j,k) + D_z^{n-1}(i,j,k)}{(\Delta t)^2} \\ & + \frac{D_z^{n+1}(i,j,k) - D_z^n(i,j,k)}{\Delta t} \\ & = a_{1(i,j,k)} \frac{E_z^{n+1}(i,j,k) - 2E_z^n(i,j,k) + E_z^{n-1}(i,j,k)}{(\Delta t)^2} \\ & + a_{2(i,j,k)} \frac{E_z^{n+1}(i,j,k) - E_z^n(i,j,k)}{\Delta t} \\ & + \sigma_{(i,j,k)} \frac{E_z^{n+1}(i,j,k) + E_z^n(i,j,k)}{2} \end{aligned} \tag{10}$$

where $a_{1(i,j,k)} = \epsilon_0 \epsilon_{\infty(i,j,k)} \tau_{\mathrm{D}(i,j,k)}$ and $a_{2(i,j,k)} = (\epsilon_0 \epsilon_{\mathrm{S}(i,j,k)} + \sigma_{(i,j,k)} \tau_{\mathrm{D}(i,j,k)})$.

As before, (9) and (10) are split into the three different directions. For instance, the y part of (9) is

$$\begin{aligned} & \frac{1}{2}\left\{ -\frac{H_x^{n+\frac{2}{3}}(i,j,k) - H_x^{n+\frac{2}{3}}(i,j-1,k)}{\Delta y} \right. \\ & \qquad \left. -\frac{H_x^{n+\frac{1}{3}}(i,j,k) - H_x^{n+\frac{1}{3}}(i,j-1,k)}{\Delta y} \right\} \\ & = \frac{D_z^{n+\frac{2}{3}}(i,j,k) - D_z^{n+\frac{1}{3}}(i,j,k)}{\Delta t} \\ & \therefore D_z^{n+\frac{2}{3}}(i,j,k) \\ & = D_z^{n+\frac{1}{3}}(i,j,k) - \frac{\Delta t}{2\Delta y} \\ & \times \left\{ H_x^{n+\frac{2}{3}}(i,j,k) - H_x^{n+\frac{2}{3}}(i,j-1,k) \right. \\ & \qquad \left. + H_x^{n+\frac{1}{3}}(i,j,k) - H_x^{n+\frac{1}{3}}(i,j-1,k) \right\} \end{aligned} \tag{11}$$

and the y part of (10) is

$$\begin{aligned}\tau_{\mathrm{D}(i,j,k)} & \frac{D_z^{n+\frac{2}{3}}(i,j,k) - 2D_z^{n+\frac{1}{3}}(i,j,k) + D_z^n(i,j,k)}{\left(\frac{1}{2}\Delta t\right)^2} \\ & + \frac{D_z^{n+\frac{2}{3}}(i,j,k) - D_z^{n+\frac{1}{3}}(i,j,k)}{\frac{1}{2}\Delta t} \\ & = a_{1(i,j,k)} \frac{E_z^{n+\frac{2}{3}}(i,j,k) - 2E_z^{n+\frac{1}{3}}(i,j,k) + E_z^n(i,j,k)}{\left(\frac{1}{2}\Delta t\right)^2} \\ & + a_{2(i,j,k)} \frac{E_z^{n+\frac{2}{3}}(i,j,k) - E_z^{n+\frac{1}{3}}(i,j,k)}{\frac{1}{2}\Delta t} \\ & + \sigma_{(i,j,k)} \frac{E_z^{n+\frac{2}{3}}(i,j,k) + E_z^{n+\frac{1}{3}}(i,j,k)}{2} \\ & \therefore E_z^{n+\frac{2}{3}}(i,j,k) \\ & = c_{1(i,j,k)} D_z^{n+\frac{2}{3}}(i,j,k) \\ & - c_{2(i,j,k)} D_z^{n+\frac{1}{3}}(i,j,k) + c_{3(i,j,k)} D_z^n(i,j,k) \\ & + c_{4(i,j,k)} E_z^{n+\frac{1}{3}}(i,j,k) - c_{5(i,j,k)} E_z^n(i,j,k) \end{aligned} \tag{12}$$

where

$$\begin{aligned} c_{1(i,j,k)} &= \frac{\tau_{\mathrm{D}(i,j,k)} + \frac{1}{2}\Delta t}{a_{1(i,j,k)} + \frac{1}{2}\Delta t \cdot a_{2(i,j,k)} + \frac{\left(\frac{1}{2}\Delta t\right)^2 \sigma_{(i,j,k)}}{2}} \\ c_{2(i,j,k)} &= \frac{2\tau_{\mathrm{D}(i,j,k)} + \frac{1}{2}\Delta t}{a_{1(i,j,k)} + \frac{1}{2}\Delta t \cdot a_{2(i,j,k)} + \frac{\left(\frac{1}{2}\Delta t\right)^2 \sigma_{(i,j,k)}}{2}} \\ c_{3(i,j,k)} &= \frac{\tau_{\mathrm{D}(i,j,k)}}{a_{1(i,j,k)} + \frac{1}{2}\Delta t \cdot a_{2(i,j,k)} + \frac{\left(\frac{1}{2}\Delta t\right)^2 \sigma_{(i,j,k)}}{2}} \\ c_{4(i,j,k)} &= \frac{2a_{1(i,j,k)} + \frac{1}{2}\Delta t \left(a_{2(i,j,k)}\right) - \frac{\left(\frac{1}{2}\Delta t\right)^2 \sigma_{(i,j,k)}}{2}}{a_{1(i,j,k)} + \frac{1}{2}\Delta t \cdot a_{2(i,j,k)} + \frac{\left(\frac{1}{2}\Delta t\right)^2 \sigma_{(i,j,k)}}{2}} \\ c_{5(i,j,k)} &= \frac{\epsilon_0 \epsilon_{\infty(i,j,k)} \tau_{\mathrm{D}(i,j,k)}}{a_{1(i,j,k)} + \frac{1}{2}\Delta t \cdot a_{2(i,j,k)} + \frac{\left(\frac{1}{2}\Delta t\right)^2 \sigma_{(i,j,k)}}{2}}. \end{aligned}$$

Substitution of (12) into (7) so as to remove $E_z^{n+(2/3)}(i+1,j,k)$ and $E_z^{n+(2/3)}(i,j,k)$ yields

$$\begin{aligned} H_x^{n+\frac{2}{3}}(i,j,k) = & - d_2 c_{1(i,j+1,k)} D_z^{n+\frac{2}{3}}(i,j+1,k) \\ & + d_2 c_{2(i,j+1,k)} D_z^{n+\frac{1}{3}}(i,j+1,k) \\ & - d_2 c_{3(i,j+1,k)} D_z^n(i,j+1,k) \\ & + d_2 c_{1(i,j,k)} D_z^{n+\frac{2}{3}}(i,j,k) \\ & - d_2 c_{2(i,j,k)} D_z^{n+\frac{1}{3}}(i,j,k) \\ & + d_2 c_{3(i,j,k)} D_z^n(i,j,k) \\ & + d_2 c_{5(i,j+1,k)} E_z^n(i,j+1,k) \\ & - \left(d_2 c_{4(i,j+1,k)} + d_2\right) E_z^{n+\frac{1}{3}}(i,j+1,k) \\ & + \left(d_2 c_{4(i,j,k)} + d_2\right) E_z^{n+\frac{1}{3}}(i,j,k) \\ & - d_2 c_{5(i,j,k)} E_z^n(i,j,k) + H_x^{n+\frac{1}{3}}(i,j,k) \end{aligned} \tag{13}$$

where $d_2 = (1/2)\Delta t / \mu \Delta y$.

Inserting (13) into (11) to remove $H_x^{n+(2/3)}(i,j,k)$ and $H_x^{n+(2/3)}(i,j-1,k)$ yields

$$\begin{aligned} & d_2 c_{1(i,j+1,k)} D_z^{n+\frac{2}{3}}(i,j+1,k) + d_2 c_{1(i,j-1,k)} D_z^{n+\frac{2}{3}}(i,j-1,k) \\ & \qquad - \left(2 d_2 c_{2(i,j,k)} + \frac{\Delta y}{\frac{1}{2}\Delta t}\right) D_z^{n+\frac{2}{3}}(i,j,k) \\ & = d_2 c_{2(i,j+1,k)} D_z^{n+\frac{1}{3}}(i,j+1,k) \\ & - d_2 c_{3(i,j+1,k)} D_z^n(i,j+1,k) \\ & - \left(2 d_2 c_{2(i,j,k)} + \frac{\Delta y}{\frac{1}{2}\Delta t}\right) D_z^{n+\frac{1}{3}}(i,j,k) \\ & + 2 d_2 c_{3(i,j,k)} D_z^n(i,j,k) + d_2 c_{2(i,j-1,k)} D_z^{n+\frac{1}{3}}(i,j-1,k) \\ & - d_2 c_{3(i,j-1,k)} D_z^n(i,j-1,k) \\ & - \left(d_2 c_{4(i,j+1,k)} + d_2\right) E_z^{n+\frac{1}{3}}(i,j+1,k) \\ & - \left(d_2 c_{4(i,j-1,k)} + d_2\right) E_z^{n+\frac{1}{3}}(i,j-1,k) \\ & + d_2 c_{5(i,j+1,k)} E_z^n(i,j+1,k) + d_2 c_{5(i,j-1,k)} E_z^n(i,j-1,k) \\ & \quad + 2\left(d_2 c_{4(i,j,k)} + d_2\right) E_z^{n+\frac{1}{3}}(i,j,k) \\ & - 2 d_2 c_{5(i,j,k)} E_z^n(i,j,k) \\ & + 2 H_x^{n+\frac{1}{3}}(i,j,k) - 2 H_x^{n+\frac{1}{3}}(i,j-1,k). \end{aligned} \tag{14}$$

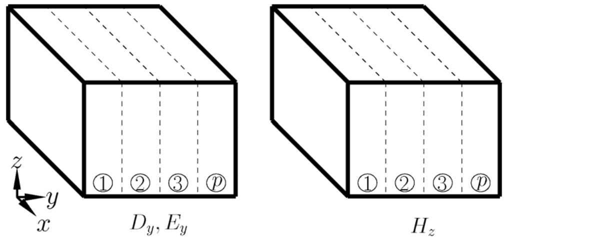

Fig. 1. Data partitioning scheme for D_y, E_y, and H_z. D_y, E_y, and H_z are all partitioned along the y-axis. Core numbers are shown in circles.

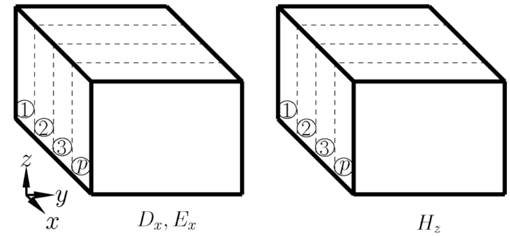

Fig. 2. D_x and E_x are partitioned along the x-axis.

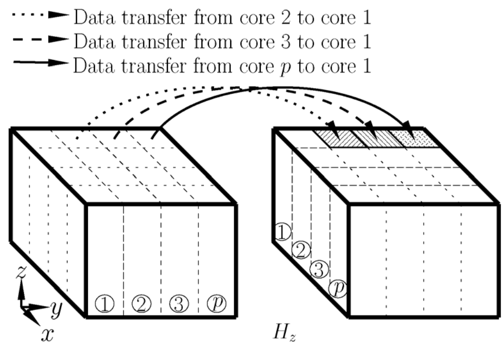

Fig. 3. Data transfers from all other cores to core 1. Arrows depict the direction of data transfer.

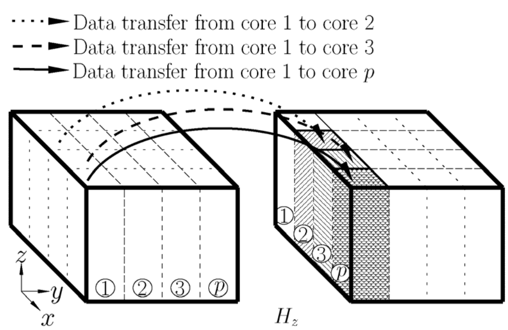

Fig. 4. Data transfers from core 1 to all other cores.

Equation (14) forms a set of simultaneous equations. Thus, $D_z^{n+(2/3)}(i,j,k)$ is obtained by solving a tri-diagonal matrix structured using (14). $D_z^{n+(2/3)}(i,j,k)$ and (12) produce $E_z^{n+(2/3)}(i,j,k)$. This newly updated $E_z^{n+(2/3)}(i,j,k)$ and (7) generate $H_x^{n+(2/3)}(i,j,k)$. The remainder of the components in each direction are derived using the same approach. The following algorithm is the complete procedure for the LOD-FDTD method with Debye media:

1) x-direction part:
   a) implicitly calculate $D_y^{n+(1/3)}$ and $D_z^{n+(1/3)}$;
   b) explicitly calculate $E_y^{n+(1/3)}$, $E_z^{n+(1/3)}$;
   c) $H_z^{n+(1/3)}$ and $H_y^{n+(1/3)}$;
2) y-direction part:
   a) implicitly calculate $D_z^{n+(2/3)}$ and $D_x^{n+(2/3)}$;
   b) explicitly calculate $E_z^{n+(2/3)}$, $E_x^{n+(2/3)}$, $H_x^{n+(2/3)}$ and $H_z^{n+(2/3)}$;
3) z-direction part:
   a) implicitly calculate D_x^{n+1} and D_y^{n+1};
   b) explicitly calculate E_x^{n+1}, E_y^{n+1}, H_y^{n+1} and H_x^{n+1}.

III. Parallelization Strategy

A. Data Partitioning Approach

Typical parallelization starts by dividing the computational domain between cores; for example, in slices along a single-space direction, as is shown in Fig. 1. In the case of the explicit FDTD parallelization, each core updates both $\boldsymbol{E}$ and $\boldsymbol{H}$ inside its slab at each time-step, afterwards sharing $\boldsymbol{H}$ at the interface between the slabs with the cores in charge of the adjacent slices, thus requiring only a one-to-one sending/receiving communication [17] on the interface planes.

When it comes to the LOD-FDTD method with Debye media more data communication and synchronization are involved, described in the following.

1) $\boldsymbol{D}$ and $\boldsymbol{E}$ Partitioning: Partitioning D_y and E_y along the y-axis works for the parallelization of the computation of D_y and E_y in procedures 1)a) and 3)a) (described in the previous section). Similarly, D_x and E_x are partitioned along the x-axis. D_z and E_z are partitioned along the z-axis.

2) $\boldsymbol{H}$ Partitioning: There are two partitioning directions for each of H_x, H_y, and H_z. For instance, H_z is calculated according to procedures 1)a) and 2)b). At procedure 1)b) H_z is obtained using D_y and E_y. Since D_y and E_y are partitioned along the y-axis, H_z also needs to be partitioned along the y-axis. The data partitioning of H_z in procedure 1)b) is depicted in Fig. 1. At procedure 2)b), H_z is obtained using D_x and E_x. Since D_x and E_x are partitioned along the x-axis, H_z also needs to be partitioned along the x-axis. Fig. 2 depicts the data partitioning of H_z at procedure 2)b). By comparing Figs. 1 and 2, it is clearly seen that the H_z space allocated to the core 1 at procedure 1)b) does not include the entire H_z space allocated to the core 1 at procedure 2)b). Thus H_z needs to be communicated between cores to carry out computations in procedure 2)b).

In summary, in this data partitioning scheme, no data communication is required for $\boldsymbol{D}$ and $\boldsymbol{E}$ but $(1-(1/m))\times 100\%$ of $\boldsymbol{H}$ of the entire FDTD space needs to be exchanged between the cores, where m is the number of cores involved in the computation.

B. Data Communication

For simplicity, only the data communication required for H_z, just after procedure 1)b), is discussed here. At the end of procedure 1)b), each core communicates with all the other cores in order to acquire updated values of H_z to update D_x and E_x in procedure 2)a).

The data transfer required to acquire all the H_z blocks needed by core 1 is depicted in Figs. 3. Fig. 4 illustrates the H_z blocks sent from core 1 to all the other cores.

The major performance driver in this data partitioning strategy is the parallelization of $\boldsymbol{D}$ and $\boldsymbol{E}$ calculations in all three direction parts. Furthermore, no memory rotation is

TABLE I
A LOOKUP TABLE FOR SCHEDULING INTERCORE COMMUNICATION IN CASE OF FIVE CORES INVOLVED

Stage	C1	C2	C3	C4	C5
1st	C5	C4		C2	C1
2nd		C5	C4	C3	C2
3rd	C2	C1	C5		C3
4th	C3		C1	C5	C4
5th	C4	C3	C2	C1	

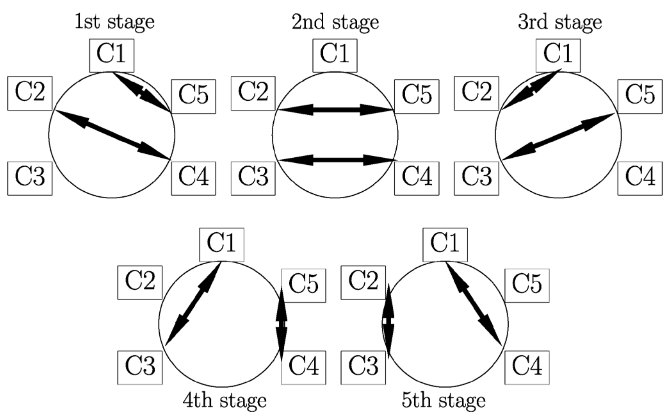

Fig. 5. One-to-one communication between cores at each stage of Table I in the case of five cores involved. C1 means core 1. The arrow between C1 and C5 means the data communication between C1 and C5. A set of these five stages is carried out at the end of procedures 1)b), 2)b), and 3)b).

required when data is transferred from one core to another. The drawback of this scheme is that all the cores in the computation have to communicate with each other in order to exchange recently updated values of $\boldsymbol{H}$, not only those on the interface plane but also those in the FDTD space, unlike the transfer in the classical FDTD method.

C. Implementation

MPI is the most commonly used method of implementing parallel algorithms on distributed memory architectures [18]. The parallel LOD-FDTD algorithm for Debye media was implemented using the Fortran programming language and the MPICH 2.0 library [23].

Instead of using the library `MPI_Alltoall` routine, our implementation involves a custom routine based on `MPI_Send` and `MPI_Recv` routines to order communication and send data between all the cores.

The custom routine involves a lookup table that is automatically generated based on the number of cores involved in the simulation. For example, when there are 5 cores for computation, Table I is automatically generated before the FDTD iteration at each core. Each core communicates with the others at the stage suggested by Table I. Table I states that there are five stages of communication at the end of procedures 1)b), 2)b), and 3)b), and the communication is depicted in Fig. 5.

IV. COMPUTATIONAL EFFICIENCY

A scalability test of the parallel LOD-FDTD method with Debye media was carried out on RIKEN's Massively Parallel Cluster (MPC), which is a part of the RIKEN Integrated Cluster of Clusters (RICC) facility. The MPC cluster consists of a total of 1048 PRIMERGY RX200S5,[1] and each node consists of two quad-core Intel Xeon processors and contains 12 GB RAM. The code was compiled using the Fujitsu Fortran compiler (configured to use the maximum level of optimization).

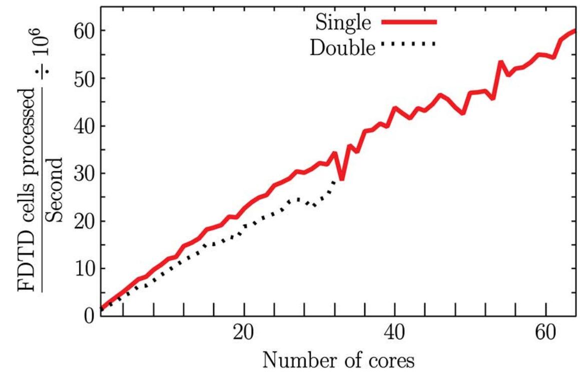

Fig. 6. Number of cells per second processed in single and double precision.

Benchmarking tests on the parallel LOD-FDTD code were conducted in both single and double precision. A computational domain of 140^3 cells per core was used to keep the computational load per core constant, independent of m. m is varied from 2 to 64.

In order to measure the performance of the MPI implementation correctly, we avoided communication between cores within a node since such communication is faster than the communication between two different nodes. Thus, only one core was used within each node. In other words, job submission was carefully tailored so that any cores, which participated in the parallel computation, could not share a motherboard.

A total of 100 FDTD time-steps were calculated in each simulation. Each simulation was repeated four times to average the elapsed time. Fig. 6 plots the number of processed FDTD cells per second as a function of the number of cores. A computational efficiency figure-of-merit $\mathcal{R}$ was defined by normalizing the computational speed to that found where only one core was used for computation:

$$\mathcal{R} = \frac{\text{cells per second using } m\text{-nodes}}{(\text{cells per second in 2 nodes})/2}.$$

Fig. 7 shows $\mathcal{R}$ of the parallel LOD-FDTD method with Debye media.

In single precision, $\mathcal{R}$ steadily and gradually deteriorates to 75% of the ideal case when the number of cores is 40 and, in double precision, 66% of the ideal case when the number of cores is 32.

We also studied how our implementation of the parallel LOD-FDTD method with Debye media compares with another implicit scheme, the alternating direction implicit finite-difference time-domain (ADI-FDTD) method [20]. Its computational efficiency had been measured by fixing the FDTD space size to 500^3 cells, recording the run time for 8 to 32 cores. In order to allow comparison between the ADI-FDTD and LOD-FDTD

[1] http://www.pcpro.co.uk/reviews/servers/354196/fujitsu-primergy-rx200-s5/specifications

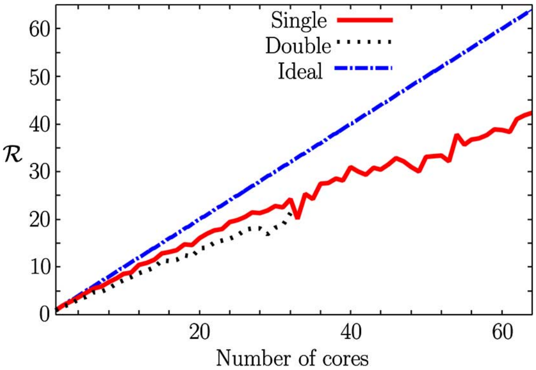

Fig. 7. $\mathcal{R}$ of the parallel LOD-FDTD method with Debye media. The "Ideal" line represents linear increase in efficiency with respect to increase in number of cores. The "Single" and "Double" lines present $\mathcal{R}$ from the single- and double-precision computation, respectively.

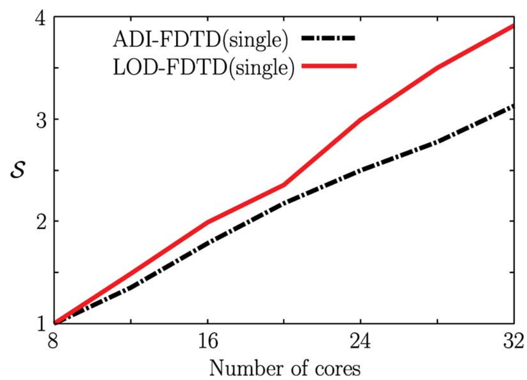

Fig. 8. $\mathcal{S}$ obtained for FDTD space size of 500^3 and up to 32 cores in case of the parallel LOD-FDTD method with Debye media and the parallel ADI-FDTD method in single precision.

methods using published data [20], we defined a speed-up factor as

$$\mathcal{S} = \frac{\text{Run time with 8 cores}}{\text{Run time}}.$$

Using our parallel LOD-FDTD code, we carried out the same simulations as [20]. Fig. 8 presents $\mathcal{S}$ for the LOD-FDTD and the ADI-FDTD methods. As the number of cores is increased, the parallel LOD-FDTD method performs better than the parallel ADI-FDTD method. This is mainly due to the fact that only $\boldsymbol{H}$ needs to be communicated between the cores. The rest of the computations, in particular the $\boldsymbol{D}$ calculations which require solution of a linear system, are performed in parallel in the LOD-FDTD method. The fact that the LOD-FDTD method has a lower communication overhead than the ADI-FDTD method also contributes to a low $\mathcal{S}$ value of the LOD-FDTD method.

We also compared the parallel LOD-FDTD method with an MPI parallel FDTD method we have developed, also including Debye-dispersive media. The parallel LOD-FDTD implementation is roughly 8.4 times slower than the parallel FDTD simulator (tested using 64 cores for a 560^3 cell problem). As a consequence, the LOD-FDTD method allows gains over the classical FDTD method, as far as we can take a CFL number N_{CFL} higher than 8.4 while maintaining accuracy under control.

TABLE II
EXAMPLE OF ONE-POLE DEBYE MEDIA PARAMETERS OF THE HUMAN TISSUES AROUND THE HEAD

Tissue	Code	σ [S/m]	ϵ_{S}	ϵ_{∞}	τ_{D} [ps]
white matter	2	0.35	41.28	24.37	33.59
midbrain	4	0.35	41.28	24.37	33.59
eyeball	5	1.45	67.71	10.31	8.27
thalamus	10	0.60	56.44	33.06	35.20
tongue	13	0.69	56.52	28.26	20.45

Regarding the accuracy, [15] involved an extensive study on the numerical dispersion of the 3D LOD-FDTD method. Ahmed *et al.* [15] showed that the error of the LOD-FDTD method rises to 5% at $N_{\mathrm{CFL}} = 10$ and 10% at $N_{\mathrm{CFL}} = 20$ when a wavelength is sampled by 100 points. Keeping N_{CFL} constant, the error decreases with increases in the spatial sampling resolution. Thus, we can set N_{CFL} as high as 20, as long as the acceptable error is above the error predicted by [15].

V. APPLICATIONS

A DBS scenario has been simulated to demonstrate a practical application of the LOD-FDTD method. DBS is a surgical operation that involves implantation of an electrode in the brain to deliver electrical stimulation to a precisely targeted area. In the treatment of Parkinson's disease, the subthalamic nucleus (STN) of a patient is stimulated by a electromagnetic field [24].

Although DBS can provide therapeutic benefits for Parkinson's disease, it has a number of risks, such as infection, skin erosion, electrode fracture, electrode dislocation, hardware failure, and associated difficulties due to the invasive electrode implantation [25]. Therefore, the application of electromagnetic wave, which may be able to provide noninvasive STN stimulation, could be an alternative method of treatment. In order to focus the electromagnetic energy on the targeted location inside the head, the waveforms on the skull, which originated from the invasive stimulation of STN, have to be known. Thus, numerical simulation of wave propagation from inside the brain to the skull is performed.

The radio environment of this practical numerical simulation was set using the digital human phantom (DHP), as in [26], provided by RIKEN, Saitama, Japan, whose usage was approved by the RIKEN ethical committee. The spatial resolution of the DHP was 1 mm in all three directions. The DHP consisted of $265 \times 490 \times 1682$ voxels and 53 distinct tissues. We fitted the one-pole Debye media parameters of human tissues [27] using the measurements provided by the United States Air force. Table II lists some of this data. The one-pole media parameters for all 53 human tissues are available in [28].

The head part above the shoulders was placed in free space, in a total domain $900 \times 900 \times 300$ cells (i.e., 90 cm $\times$ 90 cm $\times$ 30 cm), meshed with a constant space-step of 1 mm.

We placed a z-directed hard source [29] at $(i_{\mathrm{src}}, j_{\mathrm{src}}, k_{\mathrm{src}}) = (450, 450, 550)$, which corresponds to the center of the thalamus in the DHP, as shown in Table II, excited with a Gaussian pulse with spectral content up to 3.82 GHz (according to the definition given in [29]), which corresponds to a free-space wavelength of 79 mm.

Fig. 9(a) and (b) shows the excitation point with a + mark on the $x = i_{\mathrm{src}}$ plane and $z = k_{\mathrm{src}}$ plane, respectively. The eyeballs exist $10 \sim 35$ mm below the excitation point. Fig. 9(c)

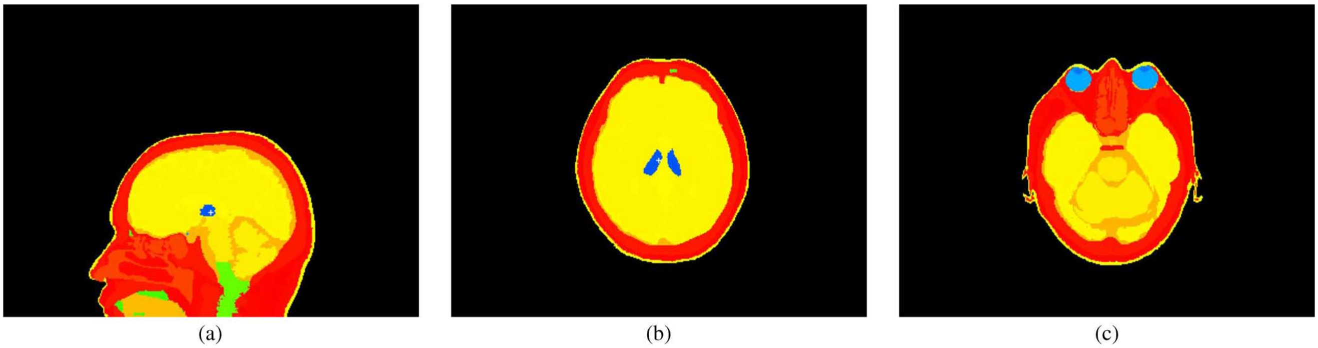

Fig. 9. Radio environment setting using the DHP and the excitation point in the FDTD space. An individual color, available in the online version, is allocated to each human tissue. (a) $x = i_{\rm src}$ plane. + shows the excitation point which is in the middle of STN. (b) $z = k_{\rm src}$ plane. + is situated in the left STN. (c) $z = k_{\rm src} - 28$ plane, which is 28 mm below the excitation location in z-direction.

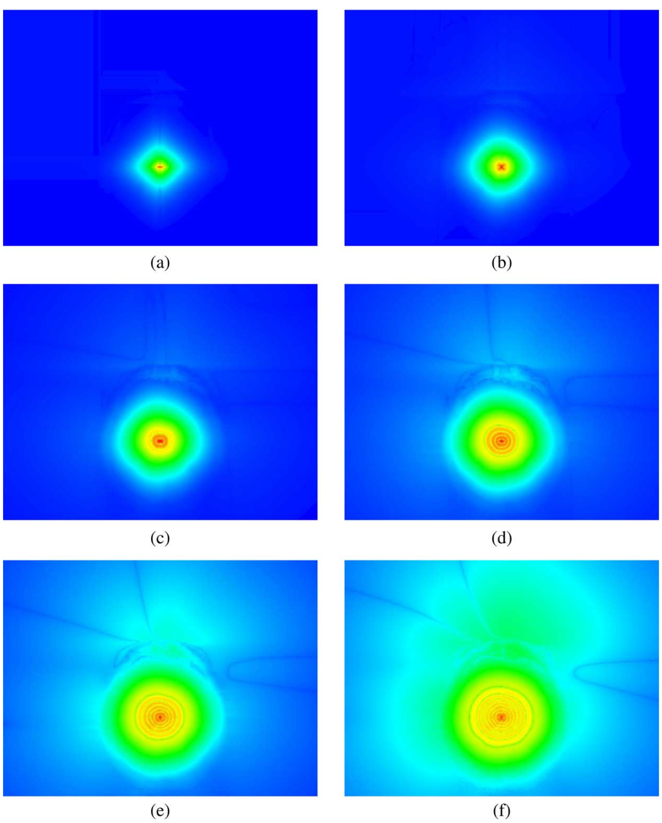

Fig. 10. Contour plot of E_z on the $z = k_{\rm src}$ plane; the computational results from the LOD-FDTD method with $N_{\rm CFL} = 20$ at $100 \cdot n \cdot \frac{10^{-11}}{3\sqrt{3}}$ seconds, where $n = 1 \sim 6$. The orientation is the same as in Fig. 9(b). They are obtained 2.4 times faster than the parallel explicit FDTD method. In the online version, the area in red has the maximum field values, and the one in blue has the minimum field values. (a) $100 \times \frac{10^{-11}}{3\sqrt{3}}$ seconds. Maximum E_z on the plane is 1. (b) $200 \times \frac{10^{-11}}{3\sqrt{3}}$ seconds. Maximum E_z on the plane is 15. (c) $300 \times \frac{10^{-11}}{3\sqrt{3}}$ seconds. Maximum E_z on the plane is 3.5. (d) $400 \times \frac{10^{-11}}{3\sqrt{3}}$ seconds. Maximum E_z on the plane is 2.4. (e) $500 \times \frac{10^{-11}}{3\sqrt{3}}$ seconds. Maximum E_z on the plane is 2. (f) $600 \times \frac{10^{-11}}{3\sqrt{3}}$ seconds. Maximum E_z on the plane is 1.6.

is 28 mm below Fig. 9(b). The DHP closes its eyes with the eyelids.

The E_z distribution on the $z = k_{\rm src}$ plane obtained from the LOD-FDTD computation with $N_{\rm CFL} = 20$ is visualized in Fig. 10. Its orientation is the same as Fig. 9(c), but the cropped area differs from Fig. 9(c). The signal comes out of eyes first and

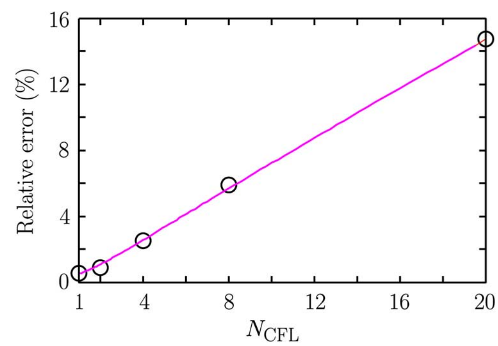

Fig. 11. Error of LOD-FDTD method relative to the explicit FDTD method when $N_{\rm CFL}$ changes from 1 to 20.

second from the left ear (due to the excitation of the left STN) and reach the $z = k_{\rm src}$ observation plane at about $300\Delta t_{\rm CFL}$ and 400 $\Delta t_{\rm CFL}$, respectively. An animation of the detailed movement of the electromagnetic wave propagation from this simulation is presented in color at http://personalpages.manchester.ac.uk/staff/fumie.costen/LODFDTDpropagation.html These results were obtained about 2.4 times faster than the in-house parallel explicit FDTD code.

We have performed the same computation, varying the $N_{\rm CFL}$ parameter between 1 and 20, and we have calculated the error of the LOD-FDTD method with respect to the usual explicit FDTD method (Fig. 11). For $N_{\rm CFL} = 8.4$, the error of the parallel LOD-FDTD method is found to be around 6%, requiring the same computational time than the parallel explicit FDTD method to reach a given physical time. For values over 8.4, the parallel LOD-FDTD method presents gains in the computational time over FDTD, linearly increasing with $N_{\rm CFL}$. For instance, for $N_{\rm CFL} = 2.4 \cdot 8.4 = 20$, the parallel LOD-FDTD takes 2.4 times less CPU time than FDTD to reach a solution, while the error becomes, as expected, 14.2%.

VI. Conclusion

The locally one-dimensional FDTD method is an alternative to the classical FDTD method, permitting us to model electrically small details with a temporal sampling larger than that governed by the CFL stability condition. In this paper, we presented an efficient parallel implementation of the LOD-FDTD method, including Debye dispersion treatment. Since it is only one order of magnitude slower than the classical parallel FDTD method, it becomes advantageous for problems where the time-step can be increased ten times or more above the FDTD stability limit ($N_{\rm CFL} > 10$), assuming that space and time sampling are set appropriately.

The performance of the parallel LOD-FDTD code was examined showing an overall good scalability, better than the parallel ADI-FDTD method, thanks to the fact that the LOD-FDTD method requires lower communications between cores than the ADI-FDTD method. The LOD-FDTD method achieves a consistent rise in performance using up to 40 cores. However, some level of saturation in efficiency is observed when more than 40 cores are utilized.

We have demonstrated the utility of this tool in a complex bio-electromagnetic problem requiring the simulation of the deep brain stimulation in the human body, densely meshed in space. The results were obtained 2.4 times faster than the parallel FDTD method using an identical computational environment.

REFERENCES

[1] A. Taflove and S. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, 3rd ed. Boston, MA, USA: Artech House, 2005.

[2] M. Livesey, J. Stack, F. Costen, T. Nanri, N. Nakashima, and S. Fujino, "Development of a CUDA implementation of the 3D FDTD method," *IEEE Antennas Propag. Mag.*, vol. 54, no. 5, pp. 186–195, Oct. 2012.

[3] M. Livesey, F. Costen, and X. Yang, "Performance of streaming SIMD extensions instructions for the FDTD computation," *IEEE Antennas Propag. Mag.*, vol. 54, no. 3, pp. 160–168, Jun. 2012.

[4] Y. Yang, R. Chen, and E. Yung, "The unconditionally stable Crank Nicolson FDTD method for three-dimensional Maxwell's equations," *Microw. Opt. Technol. Lett.*, vol. 48, pp. 1619–1622, 2006.

[5] H. K. Rouf, F. Costen, S. G. Garcia, and S. Fujino, "On the solution of 3-D frequency dependent Crank-Nicolson FDTD scheme," *J. Electromagn. Waves Appl.*, vol. 23, pp. 2163–2175, 2009.

[6] F. Zheng, Z. Chen, and J. Zhang, "A finite difference time domain method without the Courant stability conditions," *IEEE Microw. Guided Wave Lett.*, vol. 9, pp. 441–443, 1999.

[7] T. Namiki, "3-D ADI-FDTD method-unconditionally stable time-domain algorithm for solving full vector Maxwell's equations," *IEEE Trans. Microw. Theory Tech.*, vol. 48, no. 10, pp. 1743–1748, 2000.

[8] J. Shibayama, M. Muraki, J. Yamauchi, and H. Nakano, "Efficient implicit FDTD algorithm based on locally one-dimensional scheme," *Electron. Lett.*, vol. 41, no. 19, pp. 1046–1047, Sep. 2005.

[9] J. Shibayama, R. Takahashi, J. Yamauchi, and H. Nakano, "Frequency-dependent LOD-FDTD implementations for dispersive media," *Electron. Lett.*, vol. 42, no. 19, pp. 1084–1086, September 2006.

[10] J. Shibayama, R. Ando, J. Yamauchi, and H. Nakano, "A 3-D LOD-FDTD method for the wideband analysis of optical devices," *J. Lightw. Technol.*, vol. 29, no. 11, pp. 1652–1658, Jun. 2011.

[11] E. L. Tan, "Unconditionally stable LOD-FDTD method for 3-D Maxwell's equations," *IEEE Microw. Theory Wireless Compon. Lett.*, vol. 17, no. 2, p. 8587, Feb. 2007.

[12] E. L. Tan, "Fundamental schemes for efficient unconditionally stable implicit finite difference time domain methods," *IEEE Trans. Antennas Propag.*, vol. 56, no. 1, pp. 170–177, Jan. 2008.

[13] E. Li, I. Ahmed, and R. Vahldieck, "Numerical dispersion analysis with an improved LOD-FDTD method," *IEEE Microw. Wireless Compon. Lett.*, vol. 17, no. 5, pp. 319–321, May 2007.

[14] I. Ahmed, E.-K. Chua, E.-P. Li, and Z. Chen, "Development of the three-dimensional unconditionally stable LOD-FDTD method," *IEEE Trans. Antennas Propag.*, vol. 56, no. 11, pp. 3596–3600, Nov. 2008.

[15] I. Ahmed, E.-K. Chua, and E. Li, "Numerical dispersion analysis of the unconditionally stable three-dimensional LOD-FDTD method," *IEEE Trans. Antennas Propag.*, vol. 58, no. 12, pp. 3983–3989, Dec. 2010.

[16] S. G. Garcia, T. Lee, and S. Hagness, "On the accuracy of the ADI-FDTD method," *IEEE Antennas Wireless Propag. Lett.*, vol. 1, pp. 31–34, 2002.

[17] W. Yu, R. Mittra, T. Su, Y. Liu, and X. Yang, *Parallel Finite-Difference Time-Domain Method*. Norwood, MA, USA: Artech House, 2006.

[18] M. P. I. Forum, "MPI: A message-passing interface standard," Tech. Rep. [Online]. Available: www-unix.mcs.anl.gov/mpi/

[19] H. Jordan, S. Bokhari, S. Staker, J. Sauer, M. ElHelbawy, and M. Piket-May, "Experience with ADI-FDTD techniques on the Cray MTA supercomputer," in *Proc. SPIE*, 2001, vol. 4528, pp. 68–76.

[20] T. Stefanski and T. D. Drysdale, "Parallel implementation of the ADI-FDTD method," *Microw. Opt. Technol. Lett.*, vol. 51, no. 5, pp. 1298–1304, 2009.

[21] R. Joseph, S. Hagness, and A. Taflove, "Direct time integration of Maxwell's equations in linear dispersive media with absorption for scattering and propagation of femtosecond electromagnetic pulses," *Opt. Lett.*, vol. 16, no. 18, pp. 1412–1414, 1991.

[22] J. Lee and B. Fornberg, "A split step approach for the 3-D Maxwell's equations," *J. Comput. Appl. Math.*, vol. 158, no. 2, p. 485505, Sep. 2003.

[23] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel Comput.*, vol. 22, pp. 789–828, 1996.

[24] C. McIntyre, M. Savasta, L. K. Goff, and J. L. Vitek, "Uncovering the mechanism(s) of action of deep brain stimulation: Activation, inhibition, or both," *Clin. Neurophysiol.*, vol. 115, no. 6, pp. 1239–1248, 2004.

[25] M. I. Hariz, "Complication of deep brain stimulation surgery," *Movement Disorders*, vol. 17, pp. 162–166, 2002.

[26] M. Abalenkovs, F. Costen, J.-P. Bérenger, R. Himeno, H. Yokota, and M. Fujii, "Huygens subgridding for 3D frequency-dependent finite-difference time-domain method," *IEEE Trans. Antennas Propag.*, vol. 60, no. 9, pp. 4336–4344, 2012.

[27] T. Wuren, T. Takai, M. Fujii, and I. Sakagami, "Effective 2-Debye-pole FDTD model of electromagnetic interaction between whole human body and UWB radiation," *IEEE Microw. Wireless Compon. Lett.*, vol. 17, no. 7, pp. 483–485, 2007.

[28] The RIKEN webpage, Media Parameters for the Debye Relaxation Model [Online]. Available: http://cfd-duo.riken.jp/cbms-mp/ 2013

[29] F. Costen, J.-P. Berenger, and A. Brown, "Comparison of FDTD hard source with FDTD soft source and accuracy assessment in Debye media," *IEEE Trans. Antennas Propag.*, vol. 57, no. 7, pp. 2014–2022, Jul. 2009.

Tadashi Hemmi was born in Tochigi, Japan, in 1970. He received the B.E. and M.Phil. degrees in electrical engineering from the University of Manchester, U.K.

He is currently working towards the Ph.D. degree at the University of Manchester. His current research interests include computational electromagnetics.

Fumie Costen (M'07–SM'13) received the B.Sc. degree, the M.Sc. degree in electrical engineering, and the Ph.D. degree in informatics, all from Kyoto University, Kyoto, Japan.

From 1993 to 1997, she was with Advanced Telecommunication Research International, Kyoto, Japan, where she was engaged in research on direction-of-arrival estimation based on the Multiple Signal Classification algorithm for 3-D laser microvision. She filed three patents from the research in 1999 in Japan. From 1998 to 2000, she was with Manchester Computing in the University of Manchester, U.K., where she was engaged in research on metacomputing and has been a Lecturer since 2000. Her research interests include computational electromagnetics in such topics as a variety of the finite difference time domain methods for microwave frequency range and high spatial resolution and FDTD subgridding and boundary conditions. She filed a patent from the research on the boundary conditions in 2012 in the USA. Her work extends to the hardware acceleration of the computation using general-purpose computating on graphics processing units, streaming single instruction multiple data extension, and advanced vector extensions instructions.

Dr. Costen received an ATR Excellence in Research Award in 1996 and a Best Paper Award from the Eighth International Conference on High Performance Computing and Networking Europe in 2000. She was invited to give five talks in Sweden and Japan from 1996 to 2014.

Salvador Garcia (M'03) was born in Baeza, Spain, in 1966. He received the M.S. and Ph.D. degrees (with extraordinary award) in physics from the University of Granada, Granada, Spain, in 1989 and 1994, respectively.

In 1999, he joined the Department of Electromagnetism and Matter Physics, University of Granada, as an Assistant Professor (qualified for Full Professor since 2012). He has published more than 50 refereed journal articles and book chapters, and more than 80 conference papers and technical reports, and participated in several national and international projects with public and private funding. He has received grants to stay as a Visiting Scholar at the University of Duisburg in 1997, the Institute of Mobile and Satellite Communication Techniques in 1998, the University of Wisconsin-Madison in 2001, and the University of Kentucky in 2005. His current research interests include computational electromagnetics, electromagnetic compatibility, terahertz technologies, microwave imaging and sensing (GPR), bioelectromagnetics, and antenna design.

Ryutaro Himeno received the Doctor of Engineering degree from the University of Tokyo, Japan, in 1988.

In 1979, he joined Central Research Laboratories, Nissan Motor Company, Limited, Yokosuka, Japan, where he has been engaged in the research of applying computational fluid dynamics analysis to the car aerodynamic development. In 1998, he joined RIKEN, where he was the Deputy Program Director of the Next Generation Computational Science Research Program until April 2013, when he became the Director of the Advanced Center for Computing and Communication. He is also a Visiting Professor at Hokkaido University, Kobe University, and Tokyo Denki University. He currently studies computational bioengineering, high performance computing, and blood flows of human bodies.

Dr. Himeno was a winner of the Nikkei Science, Computer Visualization Contest in 2000 and the Scientific Visualization Contest in 1996, and received JSME Computational Mechanics Division Award in 1997 and JSME Youth Engineer Award in 1988. He has also received the Paper Award by NICOGRAPH in 1993 and the Giga FLOPS Award by CRAY Research, Inc. in 1990, among other awards.

Hideo Yokota received the Doctor of Engineering degree from the University of Tokyo, Japan, in 1999.

In 1993, he joined the Higuchi Ultimate Mechatronics Project, Kanagawa Academy of Science and Technology, Kawasaki, Japan. In 1999, he became the Contract Researcher of the Computational Biomechanics Unit at RIKEN, Saitama, Japan, where he was Bio-Research Infrastructure Construction Team Leader for the VCAD System Research Program, from 2004 to 2012; Cell-Scale Research and Development Team Leader for the Research Program for Computational Science, from 2007 to 2012; and has been the Image Processing Research Team Leader for the Center for Advanced Photonics since 2013. He is also a visiting Professor at Hokkaido University, Kobe University, Tokai University, and the Tokyo University of Agriculture and Technology. He currently studies biomedical imaging and image processing to the biomedical simulation.

Dr. Yokota received the Bioimaging Society's Best Image Award in 2005; the Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology, and the Young Scientists Prize in 2008.

Mehshan Mustafa, photograph and biography not available at the time of publication.