Advanced Methods for Electromagnetic Simulation

● Peter Chow ● Ross Nobes ● Tetsuyuki Kubota ● Takefumi Namiki

(Manuscript received April 1, 2007)

Computer simulation has become an integral tool in the design of modern electronic devices. Ever-increasing operating frequencies and component miniaturisation has necessitated the development of sophisticated mathematical approaches for the solution of the electromagnetic problem. In this paper we describe one such approach that uses the finite-difference time-domain method with multiple levels of grid embedding. This approach was designed for high performance on parallel computers.

1. Introduction

Computer simulation of electromagnetic behaviour is a vital part of the design of modern electronic devices. Increasing miniaturisation and higher operating frequencies, combined with a need to study complete systems where feature sizes can vary over many orders of magnitude, are placing great demands on simulation software.

A popular method for computational electromagnetic simulation is the finite-difference time-domain (FDTD) method.[1),2)] FDTD solves the time-dependent Maxwell equations using an explicit leapfrog time-stepping scheme. To ensure numerical stability, the Courant-Friedrichs-Lewy (CFL) condition[3)] limits the size of the time-step based on the smallest feature in the domain. For example, for three-dimensional models that simulate system-level details of mobile devices, the smallest features are in the nanometre (10^{-9} m) range, leading to a time-step in the attosecond (10^{-18} s) range. The feature sizes of mobile devices are in the centimetre (10^{-2} m) range, while for health and safety studies that include human models, features are in the metre range. Direct analysis of such models with the FDTD method using fine grids and small time-steps is extremely demanding in terms of memory and analysis time. In this paper we describe approaches that we and others have developed to improve the efficiency and utility of FDTD electromagnetic simulations for such multi-scale systems. All improvements are being progressively implemented into Fujitsu's commercial FDTD software package Poynting.[4)]

2. Computational approaches

2.1 Embedded-grid FDTD method

Normal FDTD methods use either a uniform or non-uniform grid as shown in **Figures 1 (a) and (b)**.

The embedded-grid (EG-FDTD) approach[5)-11)] (also called sub-gridding or multi-grid) strategically embeds fine grids in regions that require high resolution as shown in **Figure 1 (c)**. In the case shown, the embedded grid has a refinement level of three relative to the coarse grid; that is, in each spatial direction there are three fine-grid cells for each coarse-grid cell.

The advantage of EG-FDTD is that a fine grid can be used only in regions where this is needed, with substantial savings in computing

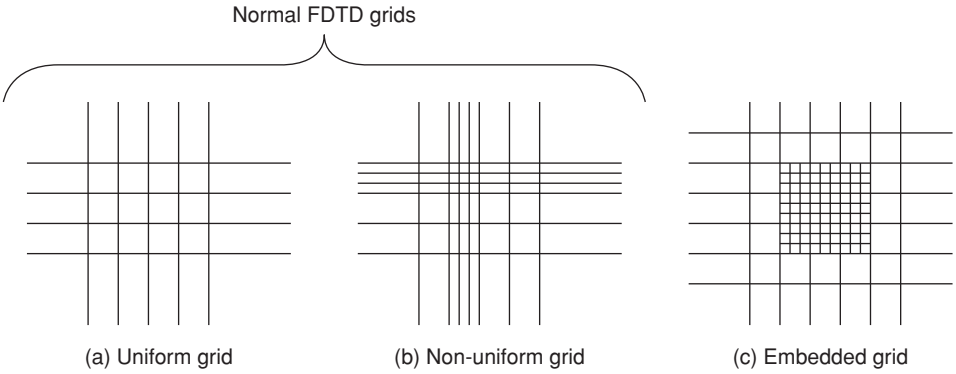

Figure 1
Comparison of grid types used in normal FDTD and embedded-grid FDTD methods.

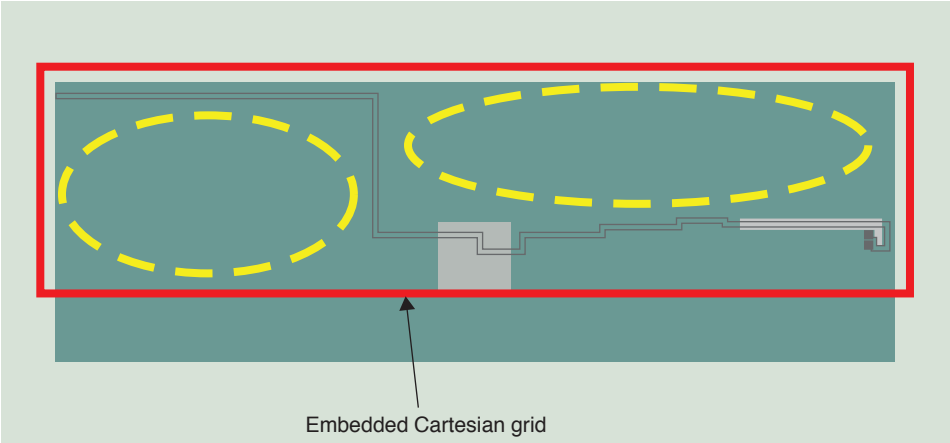

Figure 2
Use of single embedded Cartesian grid for signal-line structure.

resources. However, to avoid artefacts and ensure numerical stability, care must be taken in the procedure used to transfer values of the electric and magnetic fields between spatial grids (and between temporal grids if different time-steps are used for coarse and fine regions).

2.2 Block solver

Cartesian grids are commonly used in the EG-FDTD approach. Such grids are efficient both in terms of memory requirements and CPU processing. Their drawback, however, is that they are poor at representing irregularly shaped objects. For example, for the signal-line structure shown in **Figure 2**, a Cartesian grid is rather inefficient. The dotted ovals indicate areas where high resolution is not needed and show that about 50 percent of the rectangular Cartesian grid simply covers empty signal-line space.

A better approach is to cover the object with several smaller embedded grids or blocks as shown in **Figure 3**, which reduces the memory requirements and computational time.

However, a simplistic application of this procedure leads to an incorrect solution because of temporal disconnection between the grids/blocks. We have, therefore, implemented[12] a block-solve procedure with the correct connection

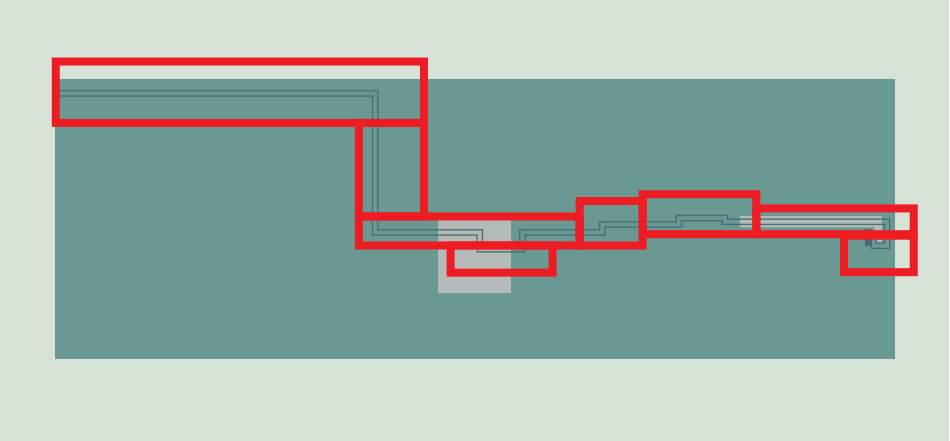

Figure 3
Block approach to EG-FDTD for signal-line structure.

Table 1
Comparison of block-solve procedure with basic embedded-grid FDTD approach.
H and E refer to the coarse grid, and h and e to the fine grid

Block-solve EG-FDTD	Basic EG-FDTD
1. Coarse-grid time loop	1. Coarse-grid time loop
2. Update H-fields	2. Update H-fields
3. Update E-fields	3. Update E-fields
4. Embedded grid a. Fine-grid time loop b. MG calculations, each grid in turn Interpolate values Update h-fields Update e-fields c. Exchange values between grids d. Return to 4a if not end time e. Map values to coarse grid, each grid in turn	4. Embedded grid a. Fine-grid time loop b. MG calculations Interpolate values Update h-fields Update e-fields c. Return to 4a if not end time d. Map values to coarse grid
5. Return to 1 if not end time	5. Return to 1 if not end time

between the embedded grids.

Table 1 compares the basic EG-FDTD procedure with the block-solve method, and **Figure 4** shows the required data exchange for a two-dimensional case.

Each grid is updated in turn, block by block. The crucial factor is the exchange of values at the grid interfaces for each time-step. The standard FDTD solver is used for each block, and the procedure does not introduce any new calculations — only data exchange at the block interfaces for each time-step.

For the signal-line structure shown in Figure 3, the block-solve procedure cuts the CPU time required by a factor of two and leads to numerical results indistinguishable from the single-block EG-FDTD approach (**Figure 5**).

2.3 Multi-level refinement

Figure 6 illustrates the multi-level EG-FDTD procedure.[13] Three levels of grid embedding are used, with each embedded grid finer than the one before. In this case, a constant refinement factor of two is used in going from one level to the next. Multi-level EG-FDTD provides a refinement process where, for both the spatial

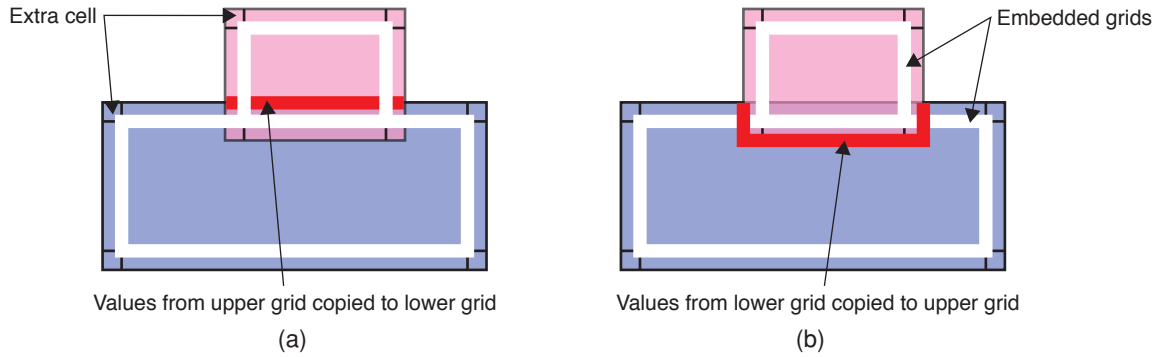

Figure 4
Layout for embedded-grid block solver. (a) Exchange of data from upper grid to lower grid. (b) Exchange of data from lower grid to upper grid. The extra cells are required for interpolation of values from the coarse grid. White rectangles indicate regions where fine grid values are mapped back to the coarse grid.

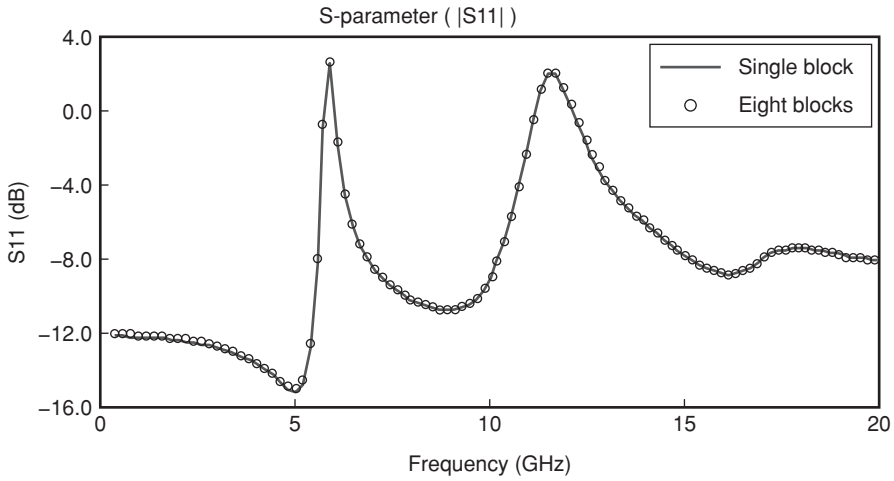

Figure 5
Calculated S-parameters for simulations with single embedded grid and with eight connected blocks. For an explanation of S-parameters, see http://en.wikipedia.org/wiki/S-parameter.

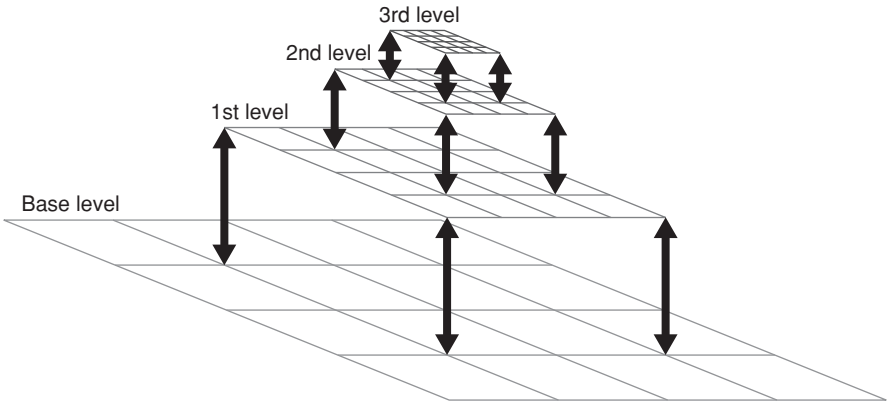

Figure 6
Multi-level EG-FDTD procedure.

and temporal spaces, the grid-cell aspect scale changes gradually rather than abruptly. Such an approach normally leads to enhanced numerical stability.

2.4 Parallelisation

Distributed-memory clusters are becoming the standard platform for technical computing. Issues such as data locality, communication between processing elements, input, and output all have to be addressed for efficient parallelisation on such distributed-memory systems. Using computer-aided parallelisation tools can significantly reduce the time required and at the same time reduces the possibility of human coding errors. We have parallelised the EG-FDTD code as implemented in our modified version of Poynting, including the block solver and multi-level refinement, using a computer-aided parallelisation tool called ParaWise.[14] The strategy adopted is to first parallelise the underlying FDTD algorithm and the input/output, followed by the embedded-grid and multi-level refinement elements, and finally to consider load balancing in more detail.

2.4.1 FDTD parallelisation

The basic FDTD procedure, which accounts for 85 percent of the code, was parallelised using ParaWise. For shared-memory systems, the parallel source code produced has OpenMP directives, while for distributed-memory systems, message passing interface (MPI) calls are inserted. At present ParaWise uses a master node model for input and output to achieve file compatibility. This means that for distributed memory systems, input is read from the master node and sent to all other nodes. For output, the reverse is applied, with all output data from the other nodes being collected by the master before writing to disk. In some cases, such an input/output model is inefficient and can lead to deterioration in performance. The method that we have implemented is for each node to output to a temporary file and to combine these together at the end to achieve file compatibility. Naturally, the combination step is not required for downstream programs such as visualisation programs that can read the parallel temporary file.

2.4.2 Embedded-grid, block-solve, and multi-level

The computer aided tool is not able to handle the embedded-grid, block-solve, and multi-level elements, and thus this parallelisation must be done manually. Fortunately, the same communications arrangement for the FDTD can be applied and greatly simplifies the parallelisation. This means the communication space is a subset of the coarse grid domain partition. **Figure 7** shows a coarse grid with nine domain partitions with two embedded grids. The coarse grid domain partition subset for the small embedded grid comprises nodes 4, 5, 7, and 8, while for the large embedded grid the nodes are 2, 3, 5, 6, 8, and 9. This is a simple change of the domain partition list before performing each embedded grid, and on finishing, the domain partition list is returned to its previous state. This mechanism is naturally applicable for the multi-level element.

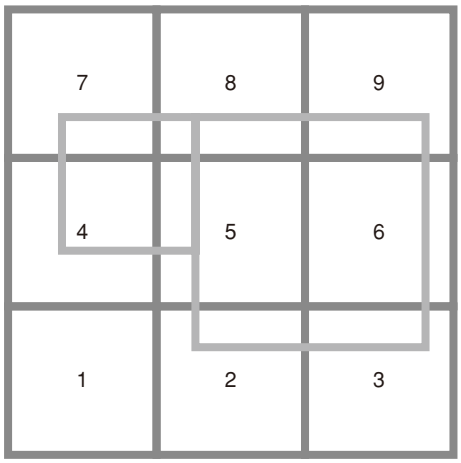

Figure 7
Data partitioning for embedded-grid parallelisation on nine nodes.

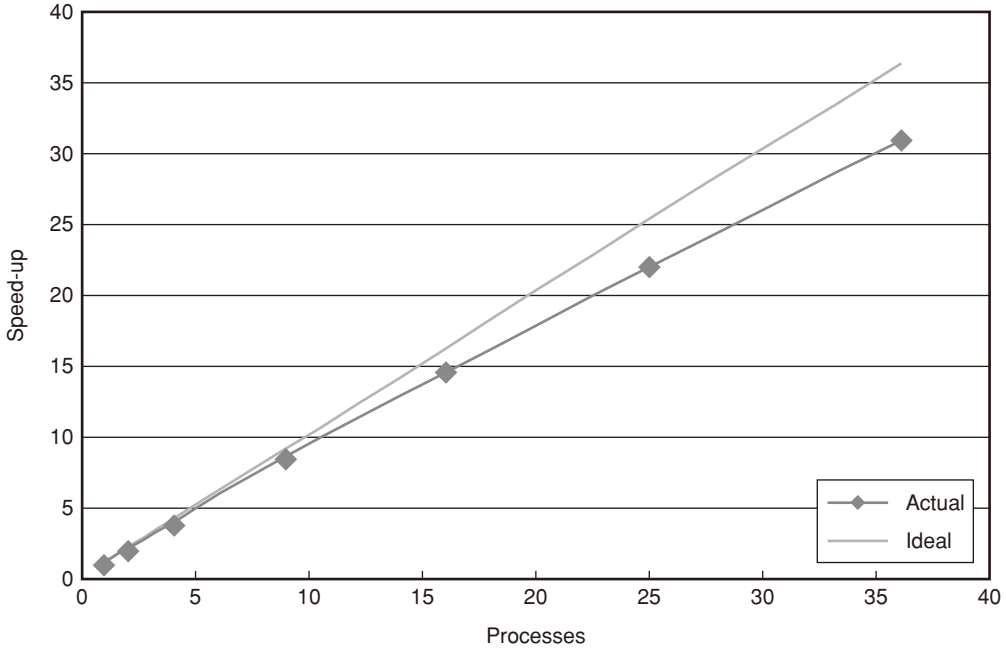

Figure 8
Parallel speed-up for Poynting FDTD simulation.

Apart from the array indexing changes required for domain partition boundaries, no other modifications are required.

2.4.3 Load balancing

For the example in Figure 7, using the domain partition calculated for the coarse grid will in general not lead to acceptable load balancing. This is because nodes 5 and 8 have the highest workload (as they are involved in two embedded grids) and node 1 has the lowest workload (as it is not involved in any embedded grid). The rest of the nodes are all involved in one embedded grid.

A more load-balanced approach is to include the embedded grid information into the calculation to give a coarse grid domain partitioning that represents a more even workload.

3. Performance

The parallel performance of the basic FDTD solver implemented in our modified Poynting is illustrated by the speed-up curve in **Figure 8**. These results were generated on a cluster of Fujitsu Siemens PRIMERGY RX200 S3 servers interconnected by an InfiniBand network and correspond to 1000 time-steps for a simulation of a mobile phone antenna near a human head (using a 301 × 301 × 301 grid).

Parallel scaling is very good, even for this relatively small case, with a speed-up of more than 30 on 36 nodes.

To illustrate the overall performance enhancement for the modifications described in this paper, we consider a simulation of a hard-disk head with flexure using the model shown in **Figure 9**. This is a demanding multi-scale simulation, with features as small as 50 nm in the read-head region. The results are shown in **Table 2**.

Using the original unmodified version of Poynting and a fine grid to capture the small-scale features in this model, this simulation runs for more than 67 days on a 2.4 GHz Intel Xeon server. Use of the embedded-grid method reduces the time by a factor of 4.7, while

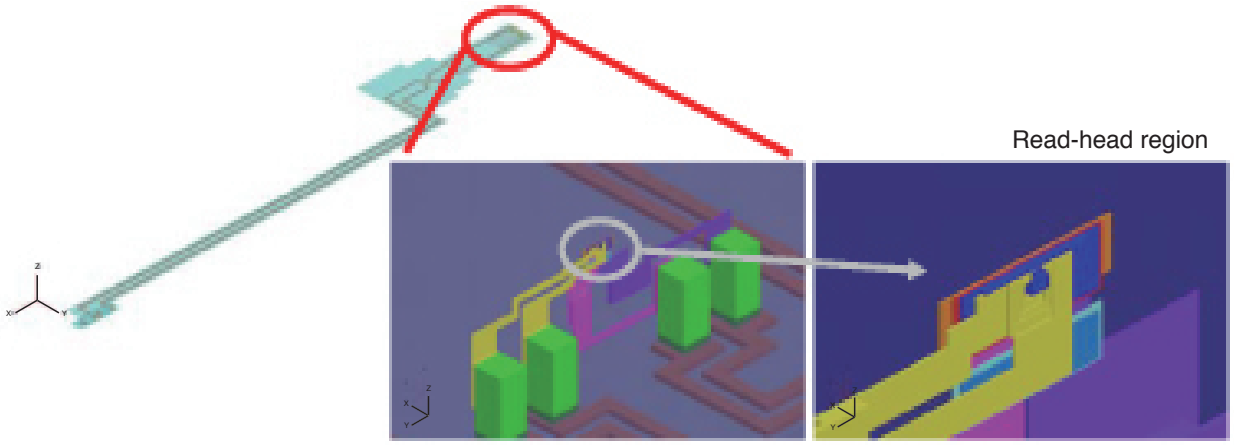

Figure 9
Model representing hard-disk head with flexure.

Table 2
Simulation time for hard-disk head with flexure.

Grid model	Serial time (days)	Parallel time (days) 16 nodes (2 × 8)
Fine grid	67.11	4.78
Single embedded grid	14.38	1.06
Three blocks	8.51	0.67
Four blocks	5.00	0.56

use of the block-solve procedure leads to a further 2.9 reduction in time. Finally, by running the calculation on a cluster of 16 Xeon servers, the time to solution is reduced to just half a day. The reduction of turnaround time for this simulation from over two months to just an overnight run clearly makes the software much more useful for device design than the original unmodified version.

4. Conclusion

In this paper we have shown that large gains in performance can be achieved for finite-difference time-domain electromagnetic simulations through a combination of grid embedding, block solvers, and parallelisation, with enhanced numerical stability achieved through a multi-level approach to grid embedding. For a multi-scale model of a hard-disk head with flexure, performance is improved by a factor of 120, even on a reasonably small compute cluster. This makes accurate simulations of the electromagnetic behaviour of multi-scale devices a reality on today's low-cost clusters. One note of caution, however, is that embedded-grid approaches, which by necessity introduce approximations through the interpolation of values between coarse and fine grids, are susceptible to numerical instability for simulations that run for many (millions of) time-steps. Future work will concentrate not only on further reductions in required memory and CPU cycles, but also on new approaches that lead to highly numerically stable EG-FDTD.

References

1) K.S. Yee: Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, **14**, p.302-307 (1966).
2) A. Taflove: Computational Electrodynamics: The Finite-Difference Time-Domain Method. Artech House, Boston, 1995.
3) R. Courant, K. Friedrichs, and H. Lewy: On the partial difference equations of mathematical physics. *Mathematische Annalen*, **100**, p.32-74 (1928).
4) Fujitsu: Poynting (in Japanese). *http://jp.fujitsu.com/solutions/plm/analysis/poynting/*
5) I.S. Kim and W.J.R. Hoefer: A local mesh refinement algorithm for the time domain finite difference method using Maxwell's curl equations. *IEEE Transactions on Microwave Theory and Techniques*, **38**, p.812-815 (1990).
6) S.S. Zivanovic, K.S. Yee, and K.K. Mei: A subgridding method for the time-domain finite-difference method to solve Maxwell's equations. *IEEE Transactions on Microwave Theory and Techniques*, **39**, p.471-479 (1991).
7) P. Monk: Sub-gridding FDTD schemes. *Journal of the Applied Computational Electromagnetic Society*, **11**, p.37-46 (1996).
8) P. Thomas and T. Weiland: A consistent subgridding scheme for the finite difference time domain method. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, **9**, p.359-374 (1996).
9) M.W. Chevalier, R.J. Luebbers, and V.P. Cable: FDTD local grid with material traverse. *IEEE Transactions on Antennas and Propagation*, **45**, p.411-421 (1997).
10) M. Okoniewski, E. Okoniewska, and M.A. Stuchly: Three-dimensional subgridding algorithm for FDTD. *IEEE Transactions on Antennas and Propagation*, **45**, p.422-429 (1997).
11) M.J. White, Z. Yun, and M.F. Iskander: A new 3D FDTD multigrid technique with dielectric traverse capabilities. *IEEE Transactions on Microwave Theory and Techniques*, **49**, p.422-430 (2001).
12) P. Chow, T. Kubota, and T. Namiki: A block-solve multigrid-FDTD method. 22nd International Review of Progress in Applied Computational Electromagnetics (ACES 2006) Conference, Miami, Florida, USA, March 2006.
13) S. Chaillou, J. Wiart, and W. Tabbara: A subgridding scheme based on mesh nesting for FDTD method. *Microwave and Optical Technology Letters*, **22**, p.211-214 (1999).
14) Parallel Software Products Inc.: ParaWise, USA. *http://www.parallelsp.com/*

Peter Chow, *Fujitsu Laboratories of Europe Ltd.*
Mr. Chow received a PhD in Computing and Mathematical Sciences from Greenwich University, London in 1993. He joined the Fujitsu Group in the UK in 1997, initially with the Fujitsu European Centre for Information Technology. Since 2002, he has been a member of the Physical and Life Sciences Research Group in the Information Systems Research division of Fujitsu Laboratories of Europe Ltd. His fields of research are Computational Science and Engineering.

Ross Nobes, *Fujitsu Laboratories of Europe Ltd.*
Mr. Nobes received a PhD in Computational Chemistry from the Australian National University in 1982. After academic positions in the University of Cambridge, UK and the Australian National University, he joined the Fujitsu Group in the UK in 1996. He currently manages the Physical and Life Sciences Research Group of Fujitsu Laboratories of Europe Ltd. His research interests are the development of scientific applications software for high-performance computers.

Tetsuyuki Kubota, *Fujitsu Ltd.*
Mr. Kubota received a doctoral degree in Plasma Physics from Kyushu University in 1998. He joined Fujitsu Ltd. in 1998, where he has been involved in the development of CAE technology. He is currently engaged in the development of electromagnetic simulation and high-speed signal transmission technology.

Takefumi Namiki, *Fujitsu Ltd.*
Mr. Namiki received the B.S. degree in Physics from Tohoku University, Sendai, Japan in 1985 and a D.E. degree in Electrical Engineering from Chiba University, Chiba, Japan in 2001. From 1986 to 1991, he was with Fujitsu Laboratories Ltd., Atsugi, Japan, where he was engaged in research of high-speed optical modulators for optical communications systems. In 1991, he joined Fujitsu Ltd., Tokyo, Japan, where he has been engaged in research and development of computational science.