Solving the Maxwell Equations with a 2D Finite Volume Method on the CM-200

Ulf Andersson*
Gunnar Ledfelt*

Abstract

The 2D transverse electric (TE) Maxwell equations have been solved by a Finite Volume method. The reconstruction of quantities on the cell interfaces is taken as an average of the values in the two neighbouring cells. We use an explicit three step Runge-Kutta method for the time-stepping. This method is well suited for a SIMD machine because of the locality of references.

We run our code on a 16K CM-200. With periodic boundary conditions the code runs in single precision at 2.5 GFLOPS, which is almost half of the peak performance. The use of absorbing boundary conditions reduces this performance with a factor of about 5. The memory size of 2 Gbyte enables us to run problems with 2048*2048 cells in double precision.

We have run the test cases C2D_2, BINA2D_4 and O2D_4b using uniform Cartesian grids. This procedure gives a saw-tooth approximation of objects. Uniform Cartesian grids were used because of the simplicity of creating them. All three cases were run on several grid sizes ranging from 512*512 to 2048*2048 cells to check the discretization error. Mesh convergence was accomplished in all three cases.

Keyword list

Transverse Electric Maxwell equations, Finite Volume, SIMD, Explicit time-stepping

*C2M2/NADA, KTH, 100 44 Stockholm, Sweden, phone: +46 8 7906333, fax: +46 8 7900930, email: ulfa@nada.kth.se and ledfelt@nada.kth.se

1 Introduction

An early and important numerical method for solution of the Maxwell equations in the time-domain was developed by Yee [12]. He used a finite-difference time-domain (FDTD) method with staggered central differencing in both space and time for a Cartesian grid. Since then, many others have used FDTD methods, most notably the group of Taflove at Northwestern university [9].

The Finite Volume method was introduced to Computational Electromagnetics (CEM) by Shankar et al. [8]. Finite Volume methods are extensively used in Computational Fluid Dynamics (CFD) and the idea of Shankar et al. was to export these methods to CEM. The Finite Volume Method, as compared to FDTD, is inherently conservative. Furthermore, complicated geometries can be naturally treated by body fitted meshes even though we do not make use of that on the test cases covered in this paper.

We have implemented the Finite Volume method on a SIMD machine, namely the CM-200. The Finite Volume or FDTD method is well suited for a SIMD machine because of the locality of references. For the time-stepping we use an explicit three step Runge-Kutta method.

2 The Finite Volume Method

In this part we will briefly demonstrate the adaption of the Finite Volume method to the Maxwell equations. For a more thorough description of the Finite Volume method see for instance [4].

The transverse electric (TE) Maxwell equations are given by

$$\left\{ \begin{array}{l} \frac{\partial}{\partial t}E_1 = \frac{1}{\epsilon}\frac{\partial}{\partial y}H_3 - \frac{\sigma}{\epsilon}E_1 \\ \\ \frac{\partial}{\partial t}E_2 = -\frac{1}{\epsilon}\frac{\partial}{\partial x}H_3 - \frac{\sigma}{\epsilon}E_2 \\ \\ \frac{\partial}{\partial t}H_3 = \frac{1}{\mu}\frac{\partial}{\partial y}E_1 - \frac{1}{\mu}\frac{\partial}{\partial x}E_2 \end{array} \right. \tag{1}$$

where $E = (E_1, E_2, 0)^T$ is the electric field, $H = (0, 0, H_3)^T$ is the magnetic field, σ is the conductivity, ϵ is the permittivity and μ is the permeability. Written in vector form they become

$$u_t + Bu_x + Cu_y + Du = 0 \tag{2}$$

where u, B, C and D are given by

$$u = \begin{pmatrix} E_1 \\ E_2 \\ H_3 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\epsilon} \\ 0 & \frac{1}{\mu} & 0 \end{pmatrix} \quad C = \begin{pmatrix} 0 & 0 & -\frac{1}{\epsilon} \\ 0 & 0 & 0 \\ -\frac{1}{\mu} & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} \frac{\sigma}{\epsilon} & 0 & 0 \\ 0 & \frac{\sigma}{\epsilon} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Integrating over a control volume Ω, independent of t, we get

$$\int\int_{\Omega}(u_t + Bu_x + Cu_y + Du)d\Omega = 0 \tag{3}$$

Defining $\overline{u}$ as the mean value of u in Ω and using Gauss theorem we get

$$A\overline{u}_t + \int_{\partial\Omega}(Bu, Cu)\cdot\hat{n}ds + AD\overline{u} = 0 \tag{4}$$

where A is the area of Ω and $\hat{n}$ is the outward normal of $\partial\Omega$. Dividing our computational domain into m polygons Ω_j and making the approximation that u is constant $= u_k$ on polygon side k we get, since (4) is valid for any Ω,

$$\frac{d}{dt}\overline{u}_j + Q_j(u) = 0, \qquad j = 1,..,m \tag{5}$$

where

$$Q_j(u) = \tfrac{1}{A_j}\textstyle\sum_{k=1}^{p}(Bu_k, Cu_k)\cdot\hat{n}_k\delta_k + D\overline{u}_j,$$

δ_k is the length of side k and p is the number of sides of Ω_j. We use quadrilateral elements, i.e. we have $p = 4$. The values u_k have to be reconstructed from the mean values $\{\overline{u}_j\}$. The easiest way is to take an average of the neighbouring cells. We use this method which is second order accurate on uniform Cartesian grids. Eq (5) is a system of ODE:s and can be solved with an ODE-solver. We use a three step Runge-Kutta method

$$\begin{aligned}
W_0 &= u^n \\
W_1 &= W_0 - \alpha_1\Delta t Q(W_0) \\
W_2 &= W_0 - \alpha_2\Delta t Q(W_1) \\
W_3 &= W_0 - \alpha_3\Delta t Q(W_2) \\
u^{n+1} &= W_3
\end{aligned} \tag{6}$$

where u^n is our solution vector after n time steps. In our algorithm we have set the parameters to be: $\alpha_1 = 1/3$, $\alpha_2 = 1/2$ and $\alpha_3 = 1$. This is second order accurate in time. An advantage of using (6) is that no intermediate solution vectors have to be stored.

The time step is bounded by (see [3] for a derivation)

$$\Delta t \leq \frac{CFL}{c}\min_{ij}\left(\frac{A}{D_{max}}\right)_{ij} \tag{7}$$

where A again is the cell area, D_{max} is the maximal diagonal of the quadrilateral cell and $c = 1/\sqrt{\mu\epsilon}$ is the speed of light. The CFL number depends on the parameters in (6). For our choice we have $CFL = \sqrt{3}$.

3 Our implementation

We have implemented the Finite Volume method on the 16K CM-200 at the Center for Parallel Computers at KTH in Stockholm. This machine has a primary memory of 2 Gbyte.

The code can handle any structured quadrilateral grid, i.e. there must be a local mapping from the physical grid to a uniform Cartesian grid. The code can handle areas with permeability, permittivity and/or conductivity different from their vacuum values.

As outer boundary condition we use an absorbing boundary condition which we will describe in the next chapter. The basic idea is to have an outer layer of cells were the Maxwell equations are modified to allow only outgoing waves. Incoming waves are treated as source terms.

At a metallic boundary the tangential component of the electric field is set to zero which is a physical boundary condition. The magnetic component is extrapolated and the normal component of the electric field is arbitrary and could for instance be set to zero.

The time step is calculated as (compare with (7))

$$\Delta t = \frac{Const}{c} \min_{ij} \left(\frac{A}{D_{max}} \right)_{ij} \tag{8}$$

where the constant is supplied by the user. For a uniform Cartesian grid where the length of the cell interfaces are δ this reduces to

$$\Delta t = Const \frac{\delta}{\sqrt{2}c} \tag{9}$$

The CM Generic Display Interface [10] is used for interactive colour visualization. This gives us the possibility of following the development of the solution and is also a great advantage when searching for bugs.

The code is written in CM Fortran which is a subset of HPF and consequently very close to Fortran 90. If we write the code to minimize the number of floating point operations we get 114 FLOP:s per Runge-Kutta step and cell. The CMSSL library V3.1 offers a stencil compiler which produces highly optimized code for computational stencils like

$$dst = w_0 * src + \sum_{i=1}^{q} w_i * cshift(src, dim = m_i, shift = n_i) \tag{10}$$

where the source array src and the destination array dst must be of the same size. The weights w_i can be floating point arrays or scalars, n_i are integers and m_i are positive integers. It is possible to have several source arrays. For further details see the manual [11].

Adjusting to this syntax we need 141 FLOP:s per Runge-Kutta step but, doing this is advantageous since the stencil compiler produces code that is more than twice as fast as the code from the standard cmf compiler. In our case, the weights in (10) depend on normals, areas and the material properties μ, ϵ and σ. The time step is also included in the weights. All these properties are constant in time which means that the weights can be precalculated. If we use a uniform Cartesian grid the weights will coincide with those of the five-point stencil commonly used in Finite Difference methods on unstaggered grids. This is not to be confused with the Yee scheme [12] which use staggered grids.

Using the stencil compiler we can, in single precision, run a periodic problem in 2.5 GFLOPS on a 16K CM-200, which is almost half the theoretical peak performance. This figure applies when calculating on the largest possible problem and is based on 141 FLOP:s per Runge-Kutta step and cell. The figure does not include initialization and post-processing.

The efficiency of the stencil compiler demands that the number of grid points in each direction is equal to a power of two. The maximum problem size that fits into the memory is 2048*2048 in double precision.

The implementation of the absorbing boundary condition is not yet optimized. At present absorbing boundary conditions take about 5 times more time than periodic boundary conditions. We estimate that this can be improved by a factor of 2. The major bottleneck is the implementation of the source term.

4 Our absorbing boundary condition

The outer boundary must be transparent for waves propagating towards the boundary from within. These waves should leave the domain without reflections. The outer boundary condition must also allow incoming waves to enter the computational domain without being disturbed by the boundary condition. This is achieved by modifying the Maxwell equations near the boundary.

The TE Maxwell equations (2) is a hyperbolic system. The matrices A and B have eigenvalues $(0, +c, -c)$. As before, c is the speed of light. These eigenvalues correspond to static solutions, right going waves and left going waves. From the theory of hyperbolic PDE we know that information is needed from the right for the left going waves corresponding to the eigenvalue $+c$ and vice versa for the eigenvalue $-c$.

If we used (2) at the outer boundary we would need information from outside of the computational domain. Instead we modify the Maxwell equations such that no information is needed from outside of our domain.

Assume that the waves scattered by the object propagate perpendicularly towards the boundary then we can, locally, formulate (2) as

$$u_t + Ku_\xi + Du = U_t + KU_\xi + DU \tag{11}$$

where $\xi = (\cos\alpha, \sin\alpha)$ is the outward normal direction of the outer boundary, $K = A\cos\alpha + B\sin\alpha$ and U is the incoming wave ($u = U + u^{scattered}$).

The eigenvalues of K are $(0, +c, -c)$. If we modify (11) by putting the third eigenvalue to zero we get a hyperbolic PDE with no left going waves and therefore we need no information from outside of the computational domain

$$u_t + K^+ u_\xi + Du = g \tag{12}$$

where $g = U_t + K^+ U_\xi + DU$. Since U is the known incoming wave the source term g is explicitly known.

The Finite Volume formulation at the outer boundary can be done by using (12) at the outermost cell layer and the ordinary Maxwell equations (2) for all inner cells.

The reconstructed values u_k needed on the outer side of the outermost cell, where (12) is used, is obtained by linear extrapolation from within. Since (12) has only outgoing waves this procedure gives a well-posed problem. At the three other cell sides u_k is obtained by taking the average of the neighbouring cells.

An important question is whether this outer boundary treatment fulfils the property of being transparent for outgoing waves and supporting the incoming waves. The incoming waves are no problem. They are perfectly introduced by the boundary condition, even for tangential incidence. The outgoing waves are more critical. If the outgoing waves are badly resolved some numerical reflections can be seen near tangential incidence. Especially the high frequency part of the wave give rise to some reflections. For example: a squared cosine pulse resolved by 64 cells give rise to 0.2 percent reflections at 45 degree incidence. If the resolution is 16 cells then the reflections increase to 5 percent. This is mainly due to the dispersion of the numerical method.

A more detailed description of this boundary condition can be found in [3].

5 Results

We have computed three test cases, namely C2D_2, BINA2D_4 and C2D_4b. For a description of these test cases see the technical description [5].

All three calculations have been made with uniform Cartesian grids. Cells having their center inside a conducting object are ignored. The field in these cells are initially put to zero and then never updated. This procedure gives a saw-tooth approximation of the objects. The great advantage of this procedure is the simplicity of creating the grid.

We have in all three cases made calculations on at least three different grid sizes to make sure that the pulses are properly resolved. In all cases we have an infinite number of frequencies present in the initial pulse. Resolving the lowest and most dominant frequency is not difficult but also some of the higher frequencies must be resolved, lest they suffer from dispersion.

The time history at a point in space is taken to be that of the cell the point lies in.

In all test cases one is asked to monitor the development in time at certain points in space. In our program this is done by checking which cell the observation points lies in and simply putting the observation point value equal to the cell value.

5.1 Cavity 2D: C2D_2

This problem is symmetric around $y_2 = 0$. We take advantage of this fact and use a rectangular computational domain of $[-0.78a, 1.78a] \times [-0.005a, 1.275a]$ with metal at $y_2 < 0$. A metallic boundary is equivalent to a symmetric boundary. The closest distance between the object and the absorbing boundary was $0.78a$. We used four different grids: 512*256, 1024*512, 2048*1024 and 4096*2048. All grids were constructed so that the height of the crack was exactly equal to $0.025a$.

Figure 1 displays the magnetic component $Z_0 H_3$ at the three observation points for the 2048*1024 grid. $Z_0 = \sqrt{\mu_0/\epsilon_0}$ is the impedance of vacuum. The results are physically reasonable. For instance we see that the reflections reach observation point 3 at $c_0 t = 1.2m$ in accordance with theory.

The 2048*1024 grid cells are quadratic with $\delta = 0.00125a$. This meant that the crack was resolved by 20*20 cells. We used a time step of $\Delta t = 3ps$ which meant that we needed 2000 iterations to reach $c_0 \Delta t = 1.8$. One iteration takes about 0.82 seconds in single precision on an 16K CM-200. This gives a total CPU time of about 27 minutes.

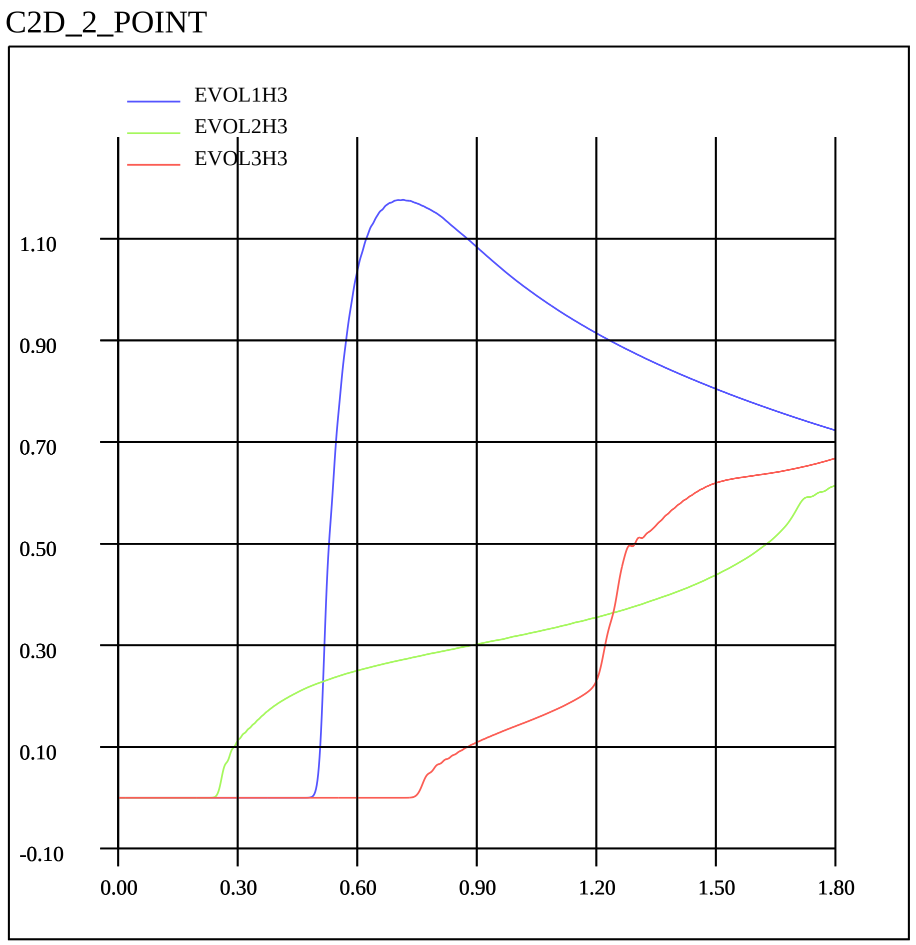

Figure 1: Magnetic component at the three observation points for the C2D_2 test case with a 2048*1024 grid.

5.2 Binaca 2D: BINA2D_4

In the BINA2D_4 test case we used a computational domain of $[-2.0a, 1.0a] \times [-1.5a, 1.5a]$. The closest distance between the objects and the absorbing boundary was a. The major reason for this large distance was the position of the observation point 2, $X_0 \approx (-1.06a, 1.06a)$.

We used three different grid sizes, namely 512*512, 1024*1024 and 2048*2048. Figure

2 displays the magnetic component Z_0H_3 at the two observation points for the 1024*1024 grid.

The 1024*1024 grid cells are quadratic with $\delta \approx 0.00293a$. We used a time step of $\Delta t \approx 6.95ps$ which meant that we needed 4800 iterations to reach $c_0\Delta t = 10$. One iteration takes about 0.39 seconds in single precision on a 16K CM-200. This gives a total CPU time of about 31 minutes.

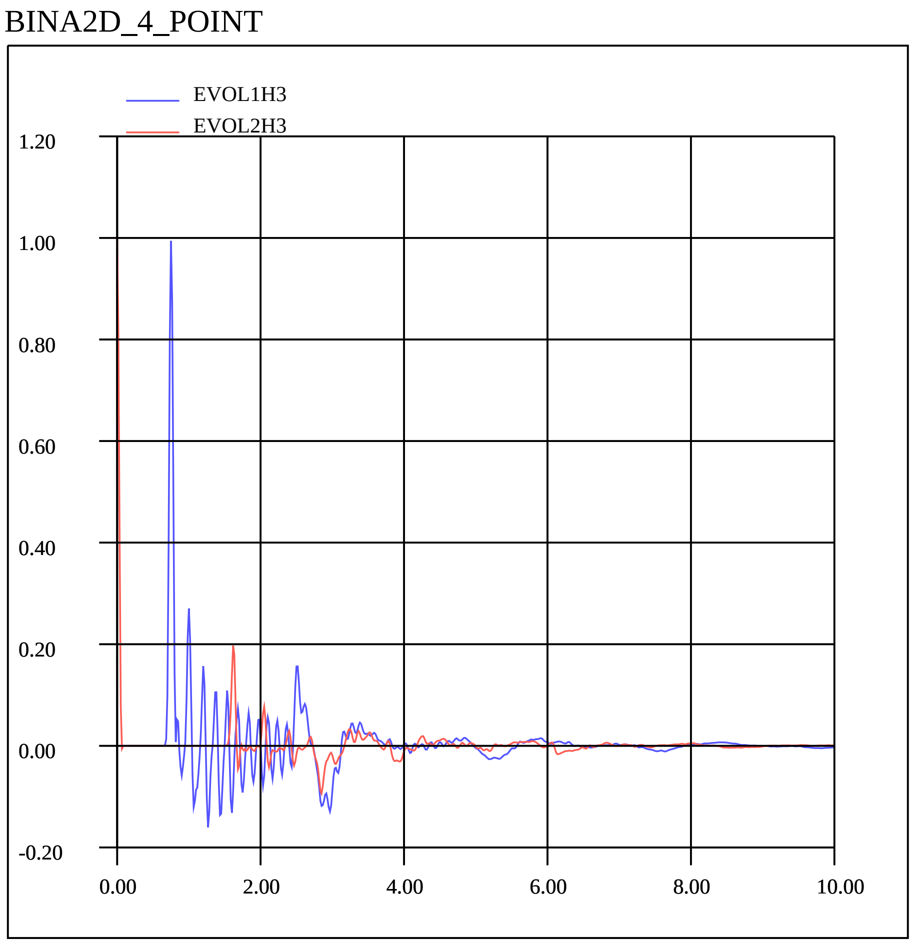

Figure 2: Magnetic component at the two observation points for the BINA2D_4 test case for the 1024*1024 grid.

5.3 Ogive 2D: O2D_4b

In the O2D_4 test case we used a computational domain of $[-1.1a, 1.1a] \times [-1.1a, 1.1a]$. The closest distance between the objects and the absorbing boundary was $0.6a$. Again, we used three different grid sizes, namely 512*512, 1024*1024 and 2048*2048.

Figure 3 displays the magnetic component Z_0H_3 at the three observation points for the 2048*2048 grid. Note that observation point 1 is situated on the surface of the Ogive.

The 2048*2048 grid cells are quadratic with $\delta \approx 0.00107a$. We used a time step of $\Delta t = 2.5ps$ which meant that we needed 4000 iterations to reach $c_0\Delta t = 3$. One iteration takes about 1.39 seconds in single precision on a 16K CM-200. This gives a total CPU time of about 93 minutes.

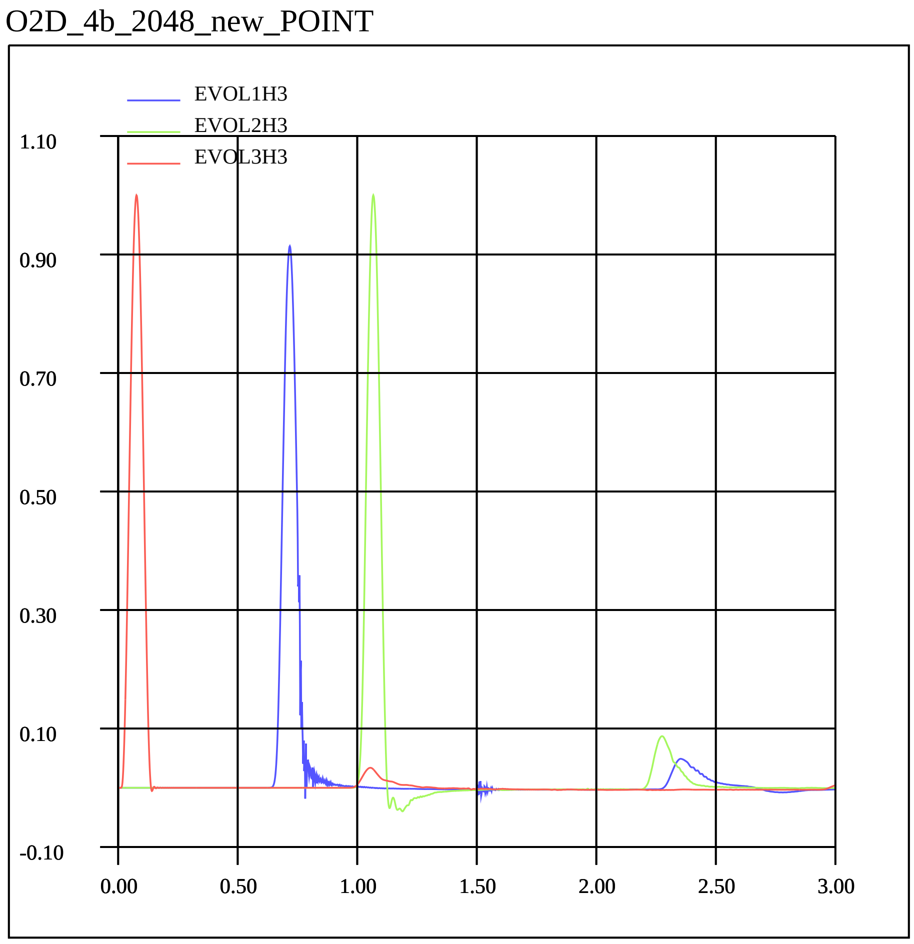

Figure 3: Magnetic component at two of the observation points for the O2D_4b test case for the 2048*2048 grid.

6 Conclusions and further development

We have shown that explicit Finite Volume codes can be successfully implemented on a SIMD machine. The introduction of absorbing boundary conditions is a severe challenge to the effectiveness of this code.

This is ongoing research and there is a lot left to be done. Among the things we wish to do are:

- Extension of the code to 3D. In 3D it will no longer be possible to have as many as a thousand cells in each direction making it necessary to devise a more effective method.

- Implementation of a Perfectly Matched Layer (PML) as outer boundary condition according to Berenger [1] and making comparisons between this, our outer boundary condition and others like for instance Engquist-Majda [2]. Some experiments have been made between Engquist-Majda and PML which indicate that PML is superior. PML is an absorbing boundary condition which in the continuous case is perfect for all frequencies and all incident angles.

- Implementation of Multi block facilities which will enable us to resolve certain areas in a better way, for instance the crack in O2D_4d.

- Adapting the code for running time harmonic problems, i.e. introducing a convergence criteria and an RCS calculation.

- Analysis of the error introduced by the saw-tooth approximation of objects. Preliminary results on a conducting cylinder indicate that the error introduced by the saw-tooth approximation is smaller than the dispersion error of the second order spatial discretisation method when one looks at the scattered field at some distance from the cylinder. We will also look into the possibility of introducing The Immersed Interface method according to Leveque and Zhang [6]. This method is used on Cartesian grids and the basic idea is to change the difference equations in nodes near an arbitrary boundary or material interface to achieve overall second order accuracy.

- Introducing higher order reconstructions (see [7]). When a wave travels a long distance dispersion of higher frequency components can become a problem. This may be improved by a higher order spatial discretisation.

- Porting the code to the IBM SP-2 at the Center for Parallel Computers at NADA, KTH in Stockholm. The IBM SP-2 presently have 55 nodes which gives 10 Gbyte of memory and a theoretical peak performance of 15 GFLOPS.

Acknowledgments

We would like to thank our advisor professor Björn Engquist for his guidance and time. We would also like to thank the staff at the Center for Parallel Computers at NADA for assisting us on using the CM-200. Finally we would like to thank Dr. Jesper Oppelstrup for his advice and encouragement.

References

[1] J-P. Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *J. Comp. Phys.*, 114:185–200, 1994.

[2] B. Engquist and A. Majda. Absorbing boundary conditions for the numerical simulation of waves. *Math. Comput.*, 31(139):629–651, July 1977.

[3] U. Gradin and G. Ledfelt. Computational electromagnetics in 2D. TRITA-NA-E9338, NADA, KTH, Stockholm, Sweden, 1993.

[4] C. Hirsch. *Numerical Computation of Internal and External Flows*, volume 1, chapter 6. Wiley, 1988.

[5] IMA (U.K.), GAMNI/SMAI (France) and Oxford Univ. Comp. Lab. (U.K.). *Technical description of Workshop on Approximations and Numerical Methods for the Solution of the Maxwell Equations*, 1995.

[6] R. J. Leveque and C. Zhang. The immersed interface method for wave equations with discontinuous coefficients. DRAFT, October 21 1994.

[7] Y. Liu. A generalized finite volume algorithm for solving the Maxwell equations on arbitrary grids. *Proc., Appl. Comp. Electromagnetics, ACES Publ.*, pages 487–494, 1994.

[8] V. Shankar, W. Hall, and H. Mohammadian. A CFD-based finite-volume procedure for computational electromagnetics - interdisciplinary applications of CFD methods. *AIAA*, pages 551–564, 1989.

[9] A. Taflove and K. R. Umashankar. Review of FD-TD numerical modeling of electromagnetic wave scattering and radar cross section. *Proc. IEEE*, 77(5):682–699, May 1989.

[10] Thinking Machines Corporation. *Generic Display Interface Reference Manual for CM Fortran, V2.0*, January 1992.

[11] Thinking Machines Corporation. *CMSSL for CM Fortran, CM-200 Edition, V3.1*, July 1993.

[12] K. S. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Trans. Antennas Prop.*, AP-14(3):302–307, May 1966.