PARALLEL FDTD ALGORITHM FOR ANALYSIS OF ELECTROMAGNETIC PROPAGATION IN BIG URBAN AREA AND IMPULSE RESPONSE

Glen RODRIGUEZ and Yasumitsu MIYAZAKI
Toyohashi University of Technology
1-1 Hibarigaoka, Tempaku-cho, Toyohashi-shi, Aichi, 441-8580, Japan
Email miyazaki@emlab.tutics.tut.ac.jp

1-Introduction

The number of user of wireless communications has been growing constantly last decade. This process has created more interest in the study of the propagation and scattering of electromagnetic waves in big areas. FDTD method is an interesting alternative to simulate those problems, but it demands a great amount of memory and computer time. Most research has been done to improve the speed by using FDTD parallel algorithms, but not to deal with the need of bigger memory than available in parallel environments, which limits the model size. This paper presents our research regarding this matter.

2-Mathematical basis of the algorithm

In this paper, a TM 2-D FDTD model is used. Yee's equations for 2-D case, with linear, isotropic, non-dispersive materials, are:

$$E_z^{n+1}(i,j) = C_a[E_z^n(i,j)] + C_x[H_y^n(i+1,j) - H_y^n(i,j)] - C_y[H_x^n(i,j+1) - H_x^n(i,j)] \quad (1)$$

$$H_x^{n+1}(i,j) = H_x^n(i,j) - D_y[E_z^{n+1}(i,j) - E_z^{n+1}(i,j-1)] \quad (2)$$

$$H_y^{n+1}(i,j) = H_y^n(i,j) - D_x[E_z^{n+1}(i,j) - E_z^{n+1}(i-1,j)] \quad (3)$$

Where C_a, C_x, C_y, D_x and D_y are constants with respect to cell (i,j) with m-th material:

$$C_a(m) = \frac{1-(s(m)\Delta t/2e(m))}{1+(s(m)\Delta t/2e(m))}$$

$$C_x(m) = \frac{\Delta t/e(m)}{1+(s(m)\Delta t/2e(m))}\cdot\frac{1}{\Delta x} \qquad C_y(m) = \frac{\Delta t/e(m)}{1+(s(m)\Delta t/2e(m))}\cdot\frac{1}{\Delta y} \quad (4)$$

$$D_x(m) = \frac{\Delta t}{m(m)\Delta x} \qquad D_y(m) = \frac{\Delta t}{m(m)\Delta y}$$

Where ε(m), μ(m) and σ(m) are the permittivity, permeability and conductivity, respectively, for material m. Inclusion of point-source antennas in the grid is done using the soft source approach. If the signal is expressed as an electric field, the equation for the E_z should be:

$$E_z^{n+1}(i,j) = C_a[E_z^n(i,j)] + C_x[H_y^n(i+1,j) - H_y^n(i,j)] - C_y[H_x^n(i,j+1) - H_x^n(i,j)] + f(n) \quad (5)$$

Where the function f is a signal expressed in the same units as E (that is, V/m). For a single point soft source, with current J_{src} dependent on time, function f becomes:

$$f(n) = \frac{\Delta t}{e} J_{src}(n\Delta t) \quad (6)$$

Yee's formulas can be expressed as a system of linear equations, as shown in eq.(7)-(8). X_i is a vector with all variables H_x, H_y and E_z in time step i, and X_0 are the initial

conditions.
Eq.(7) represents that variables at time step t+1 depend on the variables of time step t and the sources at t+1. The coefficient matrix A can be represented as a block banded matrix with two building components, identity matrix I in the main diagonal, and sparse matrix A_1, as shown in eq.(8), and the constant vector b consists of small vectors ϕ_i = [0,0,…,f_k(iΔx),…,0]', where f_k is the emission of k-th antenna at time step i. A_1 represents the dependence between fields at time i+1 and i because the medium in the simulation, and b is the source antennas.

$$X_{t+1} = A_1 X_t + f_{t+1} \tag{7}$$

$$\begin{bmatrix} I & 0 & \cdots & 0 & 0 \\ -A_1 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \\ 0 & 0 & \cdots & -A_1 & I \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_{T-1} \\ X_T \end{bmatrix} = \begin{bmatrix} AX_0 \\ f_2 \\ \vdots \\ f_{T-1} \\ f_T \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} * & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & * & 0 & 0 & 0 & 0 \\ * & * & * & * & 0 & 0 & 0 \\ * & * & * & 0 & * & 0 & 0 \\ * & * & * & 0 & 0 & * & 0 \\ 0 & 0 & 0 & * & * & * & * \end{bmatrix} \begin{bmatrix} E_z^{(1)} \\ H_x^{(1)} \\ H_y^{(1)} \\ E_z^{(2)} \\ H_x^{(2)} \\ H_y^{(2)} \\ E_z^{(3)} \end{bmatrix} = \begin{bmatrix} J^{(1)} \\ 0 \\ 0 \\ J^{(2)} \\ 0 \\ 0 \\ J^{(3)} \end{bmatrix} \tag{9}$$

In eq.(8), T is the number of time steps. Eq.(9) shows a part the matrix and vectors of eq.(8) in detail. The physical meaning of this system is better express there.
If we use eq.(2) and (3), matrix A would not be triangular, and that would mean that factorization is necessary. To avoid it, we must change eq.(2) and (3) by substituting the value of E_z^{n+1} using eq.(1). In this way, variables H_x^{n+1} and H_y^{n+1} can be calculated as linear expressions of variables at previous time step n.

3-Parallel Algorithm

A problem with the common parallel FDTD using spatial partition, as in [4][5], is the inconvenience for large problems because it has to communicate a lot of data across processors at the same time . Other problem is the need for memory; this kind of parallelization keeps the need of about 130 Gb of memory for a 2-D problem with size 100,000x100,000 cells (120 Gb for the field variables and around 10 Gb for material, boundary conditions, etc.). A different approach was used, that is, the formulation of the FDTD method like a huge system of linear equations.
First, the unknown field components will be represented as x variables, and the following notation is defined: $x_{t,i,j,c}$ represents H_x^t(i,j) if c=0, H_y^t(i,j) if c=1 and E_z^t(i,j) if c=2. The main notation, with an unique index L , will be x_L which is equivalent to some $x_{t,i,j,c}$, where L=(t-1)(3pq)+3qx+3y+c; this formula works in both ways.

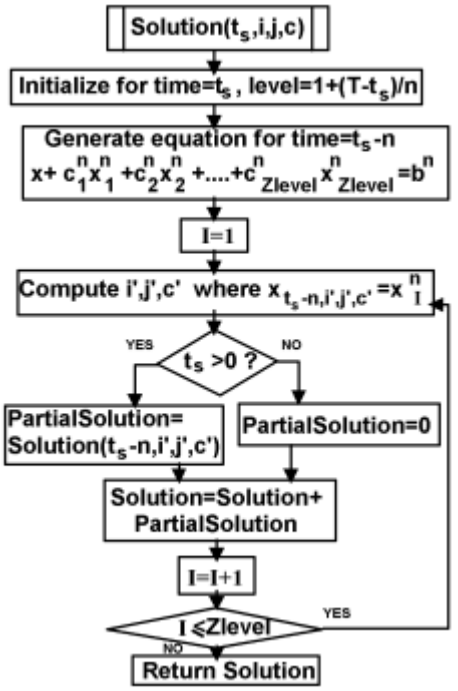

Fig.1: Flowchart of Algorithm

The sparse equations (the rows of A) are represented by arrays, containing information about the column, the row and the coefficient (value in A or any related matrix). These arrays are big and are stored across the processors. This special storing for sparse matrix does not keep any zero (most of the elements of the matrix are zeros).
Gaussian back-substitution is carried from the equation for x_L=$x_{T,i0,j0,c0}$(greatest time step

T) many times, changing original eq.(10), with x_b^a equivalent to some variable $x_{T\text{-}a,i,j,c}$ and c_b^a constant coefficients, into eq.(11).

$$x_L + c_1^1 x_1^1 + c_2^1 x_2^1 + \ldots + c_{Z1}^1 x_{Z1}^1 = b_L^1 \quad (10)$$

$$x_L + c_1^2 x_1^2 + c_2^2 x_2^2 + \ldots + c_{Z2}^2 x_{Z2}^2 = b_L^2 \quad (11)$$

Where all x_i^2 are variables in time step T-2. It must be noted that the number of terms in the last equation is greater than in the first: $z_{k+1} > z_k$. This process is carried many times, until the equation becomes:

$$x_L + c_1^n x_1^n + c_2^n x_2^n + \ldots + c_{Zn}^n x_{Zn}^n = b_L^n \quad (12)$$

For some n, the size of the array (order $6n^2$) becomes very large. Then we take the independent variables x_i^n, at time step T-n, of eq.(12), one by one, creating a new process of back substitution for this variable. The variables became a new x_L in a separate array of memory. A flowchart of the algorithm called "solution" is shown in Fig.(1).
The parallelization is done by distributing the variables in the sparse arrays between the different processors. Each processor does its respective sub-summation, as seen in eq.(13). Finally, the algorithm gets the partial summations and gives the solution for x_L.

$$x_L = -\sum_{r=1}^{p}\left(\sum_{k=1}^{z_n^p} c_k^n x_k^n\right) + b_L^n \quad (13)$$

With single precision floating-point variables, total basic memory use is in the order of 24Tn. For simulations as large as 100,000x100,000 cells, a reasonable duration of the simulated wave propagation should be 10^6 time steps. This algorithm needs around 4 Gb of memory. For the small model of next section (100x100cells), normal FDTD requires 120 Kb of memory. Our algorithm uses a level with n=30 time steps, with one level, then we need 70 kb of memory. This amount of memory for the small model is 70/120=58.3% of the memory required by the common parallelization of the FDTD method, but it is only 4/130=3.08% of the memory for the big model of size 100,000x100,000. The principle in which this method is based is to "decrease memory by increasing the computational bandwidth" (See [6]).

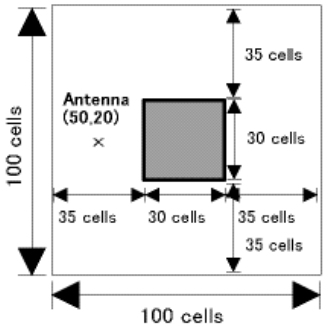

Fig.2: Small model for validation

4-Numerical Results

We have simulated a small model as fig.(2) shows (a 100x100 space, with a small solid block made of concrete, in the middle) using normal FDTD and our algorithm and compared the results, as can be seen in Fig.(3). Frequency of source= 950 MHz, cell size $\Delta x = \Delta y =$ 0.02 m, time increment Δt= 37.7 ps, relative permittivity (concrete): 3.0, conductivity of concrete, σ: 0.005 S/m, J_{max}= 1000.0 A/m, $J_z^{i,n} = J_{max} \sin(2\pi ft)$ at antenna position and $t<30\Delta t$. The results of Fig.(3) agree very well in all the values of E_z, because our algorithm neither modifies nor made simplifications of the Yee formulas.

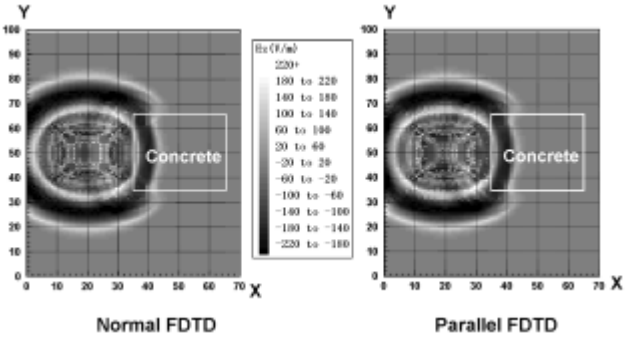

Fig.3:Comparison for E_z at $t=60\Delta t$

The objective for the continuation of this research is to simulate big models with the similar parameters as the small one, the same kind of dielectric and rectangular shapes.

5-Linear Equations and Impulse Response

The objective now is to improve the speed of the algorithm. A discrete version of the impulse response can be calculated by using a unitary impulse at t=1. In eq.(8), ϕ_1 =[0,0,..,0,1,0,...,0]' and ϕ_2 =ϕ_3 = ... =ϕ_T=0. The kernel or Green function g(r,t,r',t')=g(i,j,t), with r=(i,j), r' and t'=1 fixed, results from a FDTD simulation, and for any general source J=f(t)=f(kΔx):

$$E_z(i\Delta x, j\Delta y, k\Delta t) = \sum_{n=1}^{k} g(i, j, n)\, f(k+1-n) \qquad (14)$$

The numerical values of g are the coefficients c_i^n of the linear equations shown in eq.(12). In this equation, b_L^n=0 if there is no antenna in the vicinity of X. This is a realistic assumption for urban communication models. There is 3 different kinds of g functions, g_1, g_2 and g_3, for the influence of each component H_x, H_y and E_z , as shown in Fig.(4).

The values of g(i,j,t) depends only on the materials and sources around variable X, in a vicinity of "t" cells. The main material in an urban communication channel is free space, except the walls of the buildings. The refore, most of the values of g(i,j,t) and c_i^n will be the same. At the start, we calculate g_1, g_2 and g_3 for a free space environment with no sources and we store it in memory. This process is done only once, using the standard FDTD method. The improvement into the algorithm is shown in fig.(5).

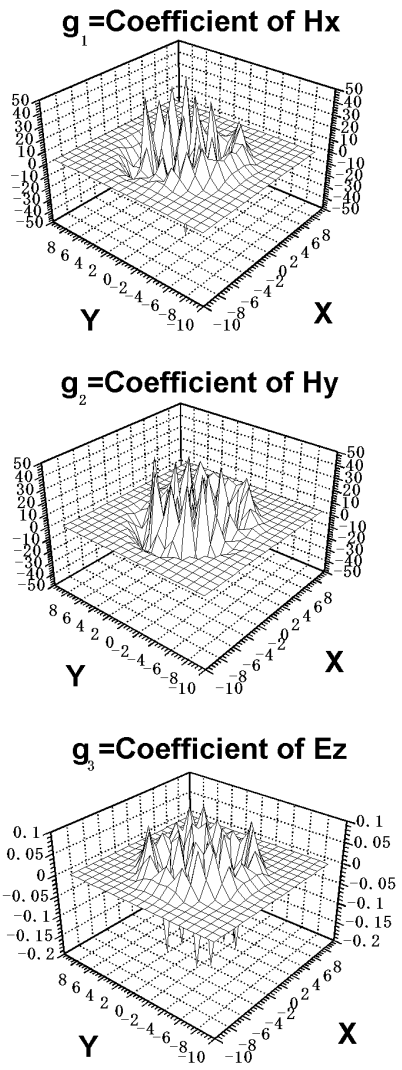

Fig.4: Kernel functions g_1, g_2, g_3(r'-r) for some E_z when t'=t-10

6-Conclusions

A new parallel 2D FDTD algorithm for the simulation of big areas has been presented. It is important to remark that our algorithm, regarding the formulas, is exactly equivalent to the Yee formulation, but expressed as matrix operations and with an additional process to change the matrix into a triangular one.

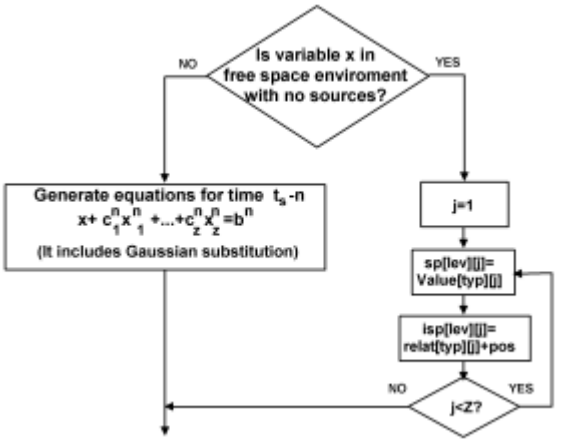

Fig.5: Improvement to algorithm

References:

[1] G.Rodriguez and Y.Miyazaki, Trans.IEE Japan, vol.121-C, No.12, 2001, pp.1826-1833.

[2] P.Selormey, Y.Miyazaki, EMCJ99, 1999, pp. 71-78.

[3] A.Fijany, M.Jensen and others, IEEE Trans. Ant. & Prop. Vol.43 N.12, Dec.1995, pp.1441-1449

[4] U.Anderson, Proc. 4th International Workshop of Applied and Parallel Comp. '98, LNCS, 1998, pp.12-19

[5] Z.Liu et al, IEEE Ant. & Prop. Magazine, Vol.37 N.5, Oct.1995, pp.64-71

[6] E.Miller, Proceedings of the IEEE, Vol.79, No.10, Oct.1991, pp.1493-1504