Introduction to the Segmented Finite-Difference Time-Domain Method

Yan Wu, Ian Wassell

Computer Laboratory, University of Cambridge, Cambridge, UK CB3 0FD

Abstract — In order to estimate path loss in various infrastructure types: tunnels, water distribution networks and bridges, we have chosen the well known Finite-Difference Time-Domain (FDTD) technique [1] due to its accuracy, flexibility and potential for visualizing the simulation results. However, problems occur owing to the high memory requirement and heavy computational burden when dealing with these large-scale systems using this technique. Following our previous work on the unique correction factor, which enables us to transform a simply structured 3D FDTD problem into a 2D simulation using the Modified 2D FDTD method [2], in this paper, we propose the Segmented FDTD (SFDTD), which divides the problem space into segments so that the computational redundancy is reduced. This technique also facilitates data reuse that eases the inconvenience imposed by configuration changes at a later date.

***Index Terms*— FDTD methods, Large-scale computing.**

I. INTRODUCTION

The finite-difference time-domain (FDTD) technique is one of the key simulation tools in the study of Electromagnetic propagation. When one twentieth of the signal wavelength is used as the basic element dimension i.e., unit cell size, good accuracy can be achieved in a FDTD simulation [3]. When conventional FDTD is applied to model large-scale problems with high signal frequencies e.g., 2.40GHz, it becomes extremely computationally demanding in terms of memory and CPU execution time. Some of the typical simulation problems of interest to us include the water distribution network which requires that wireless sensors be located in fire hydrants having an average spacing of 105m [4] or in tunnels with a typical diameter of 4~5m and lengths of 100~1000m. To solve the large-scale problem in the conventional FDTD method, non-uniform FDTD technique has been proposed, where non-uniform cells are used to form the problem space to ease the computational burden [5]. Most commercially available FDTD software also provides geometric theory of diffraction (GTD) and Ray Tracing techniques to get around the downside of the conventional FDTD [6]. In [7], a modified version of the 3D FDTD method has led to a more memory-efficient formulation, where only four field components are stored in the whole domain, with a direct memory reduction of 33% in the storage of the 3D electromagnetic fields. Most recently, considerable research has been conducted concerning Parallel Computing using the Message Passing Interface (MPI) [8]. However massively increasing computational hardware may not always be cost effective. In [2], we have demonstrated that by reducing 3D problems to 2D, large-scale problems can be addressed using regular personal computers (PCs). In this paper we present the SFDTD method, which further reduces the computational requirements and enhances the feasibility of running these simulations on a PC. For reason of simplicity, our discussion is focused in a 2D domain though it could be extended to address a full 3D situation. We will begin with a description of the existing problem, and then discuss how the proposed SFDTD method may be applied to a conventional FDTD problem. We then undertake the performance validations of the SFDTD method using the Plane Earth Model as an example. Finally we present the conclusions.

II. PROBLEM DESCRIPTIONS

Our simulation is benchmarked using a 3.46GHz, 8GB RAM Dell Precision PWS 380 computer. We assume that we are dealing with a 2D FDTD simulation and the problem space is of the dimension $IE \times JE$, where JE is fixed to be 1000 unit cells. According to the Courant Condition [9], which governs the essential stability of the FDTD method, we assume that the signal takes two time steps to travel through one unit cell in a 2D simulation, i.e.

$$\Delta t = \frac{\Delta x}{2 \cdot c_0}, \tag{1}$$

where Δt, Δx and c_0 are the duration of each time step, the dimension of each unit cell and the speed of light in a vacuum respectively. Fig.1 illustrates the exponentially increasing relationship between the size of a problem space and the computational execution time using the conventional FDTD method. The memory usage in Fig. 2 is seen to increase linearly with the length (IE) of the problem space. Note the electromagnetic field evolves with time, consequently in a large-scale FDTD simulation, a large amount of computational power is wasted updating and calculating zero values before the signal reaches the distant unit cells while in addition, a huge amount of memory is required to hold all these zeros. In order to introduce the SFDTD method, consider a problem space ($IE \times JE$) of dimension of 2.4×10^5 by 10^3. This space can be divided into 24 individual FDTD simulations (or 'segments') each of size 1.0×10^4 by 10^3. We will show that this results in a large reduction in computation time and memory usage. The detailed description of the SFDTD implementation will be presented in the next section.

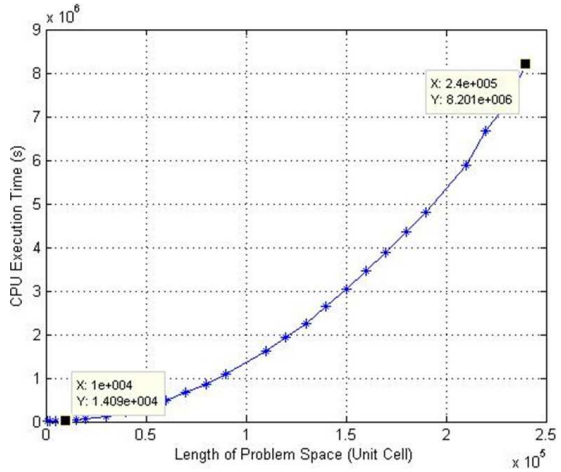

Fig. 1. Problem Space vs. CPU Time

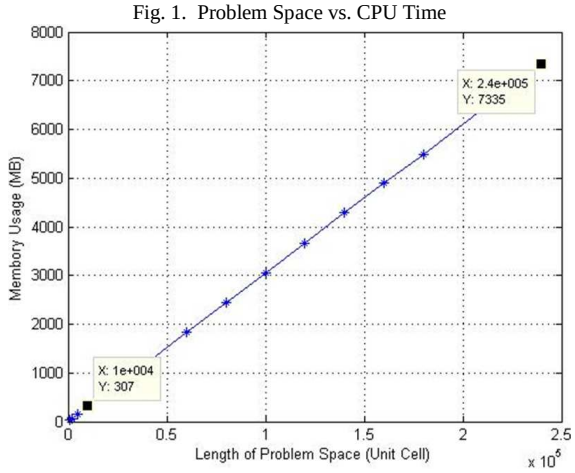

Fig. 2. Problem Space vs. Memory Usage

III. SFDTD METHOD

For now let us ignore the overheads due to the space taken up by other factors, e.g., the absorbing boundaries. The following procedures are applied to realise the SFDTD concept that is also illustrated in Fig. 3:

1. Start the conventional FDTD iteration in Segment One with the signal source S_0.
2. When Segment One reaches its steady state, i.e., all the multipath signals have arrived at each individual cell, then record 200 samples: (i) At each unit cell on Interface One. (ii) At the points of interest for the path loss investigation.
3. Save signals of length one-wavelength (i.e., one cycle in time domain) from each unit cell recorded at the 2(i) interface then form the interface array source S_1. Note that the extraction of the array source must take place at the same sampling point in time; otherwise phase information will be lost. The reason for saving one complete wavelength of samples is to maintain the signal wave's continuity.
4. Synchronously propagate the extracted interface array source at each corresponding unit cell in Segment Two.
5. Follow the same steps to complete the simulations in Segments 2, 3, 4 and up to 24 for this example.

Fig. 4.a shows the recorded samples at one unit cell on an interface. The unit cell size in our case is one twentieth of the wavelength and each unit cell requires two time steps for the signal to cross (defined by the Courant Condition). Hence each wavelength needs 40 time steps (samples) in order to be completely reconstructed and ready to be propagated in the next segment. We are also able to tell if a segment has reached its steady state by observing these samples. For example, Fig. 4.b shows the recorded samples before steady state is reached.

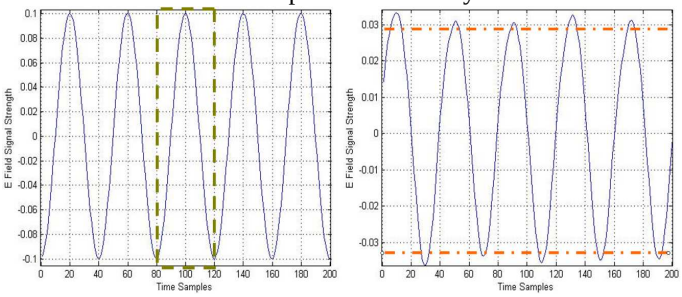

Fig. 4. Recorded samples of an interface cell

The computational time for the example using a conventional FDTD method (2.4×10^5 cells $\times$ 1 segment) would take around 95 days and use over 7.16GB memory to store the data; while the SFDTD method (1.0×10^4 cells $\times$ 24 segments) only takes about 3.9 days with a memory usage of 307MB.

IV. PLANE EARTH PATH LOSS MODEL

Now we are going to validate our SFDTD method by investigating the signal path loss for horizontally polarized antennas in a plane earth environment at a frequency of 868MHz for a maximum antenna separation of 200m while both transmitter and receiver antennas are at a height of 2m. The ground is assumed to be perfectly conducting (i.e., metal).

The well-established analytical formulation for the plane

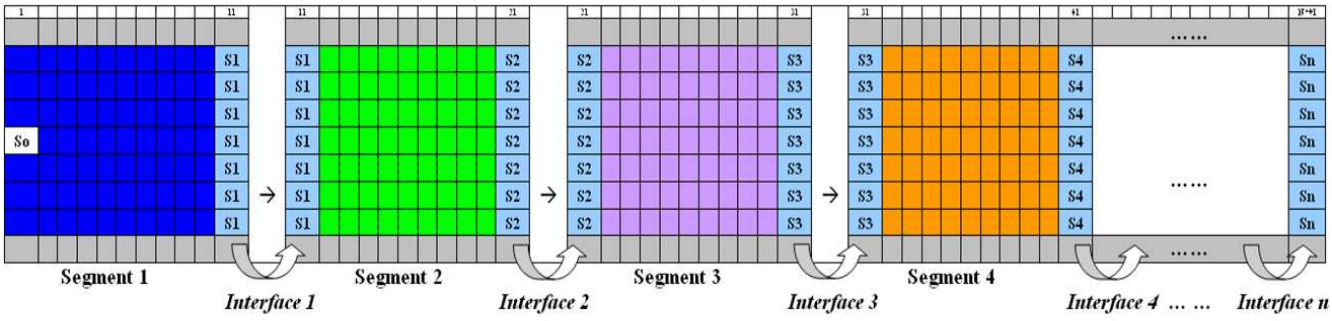

Fig. 3. Segmented Problem Space in SFDTD Simulation

earth path loss model in decibels [10] can be expressed as:

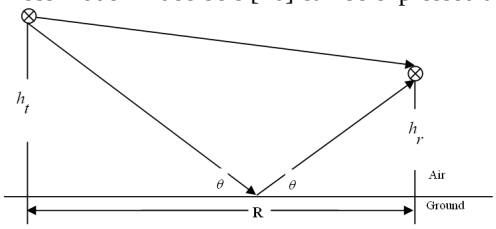

Fig. 5. Plane Earth Model

$$PL_{PE} = 10\log_{10}\left\{\left(\frac{\lambda}{4\pi R}\right)^2\left|1+\rho\exp\left(jk\frac{2h_t h_r}{R}\right)\right|^2\right\}, \quad (2)$$

where ρ is the reflection coefficient for the reflected ray; h_t and h_r are the heights of the transmitter and receiver antennas respectively; k is the free space wave-number $2\pi/\lambda$ where λ is the wavelength of the transmitted signal; For our example, ρ in the horizontal polarization can be expressed as:

$$\rho_{HP} = (\sin\theta - \sqrt{(\varepsilon_r - jx) - \cos^2\theta}) \Big/ (\sin\theta + \sqrt{(\varepsilon_r - jx) - \cos^2\theta}), \quad (3)$$

where $x = 18\times10^9\cdot\delta/f$; ε_r is relative permittivity of the ground; δ is conductivity of the ground; θ is the angle between the incident wave and the ground surface and f is the transmit frequency.

V. Performance Validations

Before we actually perform any simulations in the example plane earth environment, we need to firstly determine the total number of time steps that are required to achieve steady state. Obviously the more time steps that we iterate, the more probable it is that the problem space is going to reach steady state. The precise exactly number of total time steps required differs from environment to environment not only owing to the different distances of interest, but also the different multipath effects in particular environments. Bearing this in mind, we defined the time steps for each individual segment in this model as: 8 times the ratio between the distance of interest and the cell dimension Δx (for short, we call it the 8*ratio time scheme). Secondly, the distance to the absorbing boundaries needs to be minimised. Intuitively, we want to ensure that the points of interest are as far as possible from the absorbing boundaries so that the effects of reflections can be minimised. Our investigations have concluded that it is the JE dimension (vertical direction in Fig. 5) that has the most significant effect on the simulation results while the IE dimension (along the distance of interest, i.e., the horizontal direction in Fig. 5) has little effect in this scenario. Following our initial simulations regarding this issue, we set JE to 4000 unit cells and set the dimension of IE to be just sufficient to hold the absorbing boundaries and a distance of 200m unit cells equivalent.

Therefore the corresponding 2D FDTD problem space is set with JE equal to 4000 unit cells and segment lengths of 5m, 10m, 25m, 50m, 100m or 200m unit cells equivalent respectively. The simulation results plotted on a log scale are shown in Fig. 6, where the inset panel gives an enlarged view of the distortions owing to the different choices of segment size.

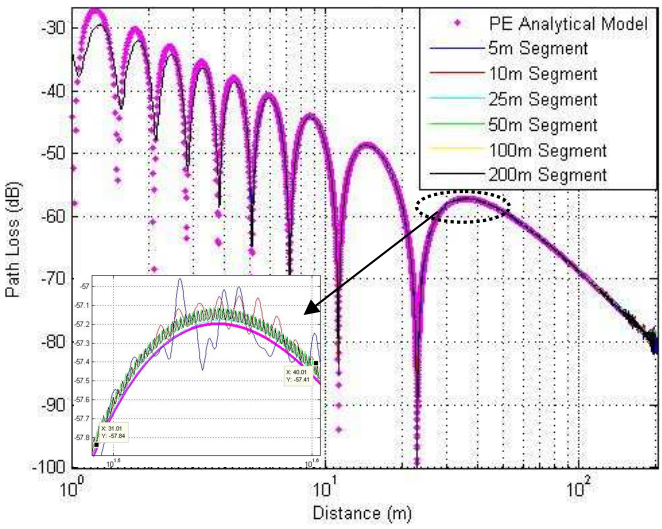

Fig. 6. Preliminary SFDTD Simulation Results

It can be seen that, in general, the SFDTD method produces high accuracy results at close ranges regardless of the segment size chosen, but shows more variability at longer ranges, particularly for smaller segment sizes. Even so it can be seen that the SFDTD simulation results fluctuate closely about the analytical solution. To tackle this problem we apply a moving average based filtering technique. This can be seen to reduce the amplitude of the ripples and a very good fit for the various segment sizes is achieved as shown in Fig. 7.

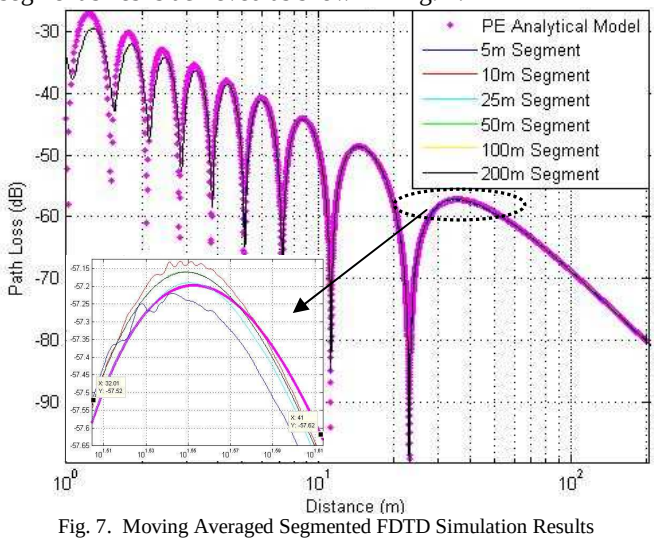

Fig. 7. Moving Averaged Segmented FDTD Simulation Results

Table 1 summarises the performance of the SFDTD method in term of computational time and memory usage. Even so, it turns out that we cannot infinitely decrease the segment size for better computational performance. For example, in Fig. 8, we observe that the use of 2m segments gives rise to huge instability. This is because the total number of time steps (8*ratio time scheme) that we iterate in each segment is not sufficient for the segment to reach its steady state. To fix the problem and so achieve stability in the SFDTD simulation for the 2m segment size, our time scheme

is altered by increasing the total number of time steps from an 8*ratio time scheme to 20*ratio time scheme as seen in Fig. 8.

	CPU Time (hrs)	Memory Usage (MB)
200m × 1 segment	55.170	1,474
100m × 2 segment	28.646	757
50m × 4 segments	22.590	399
25m × 8 segments	8.745	214
10m × 20 segments	4.877	115
5m × 40 segments	3.660	77
2m × 100 segments	3.350 (unstable) 8.375 (stable)	60

TABLE 1. SEGMENTED PLANE EARTH MODEL PERFORMANCE COMPARISONS

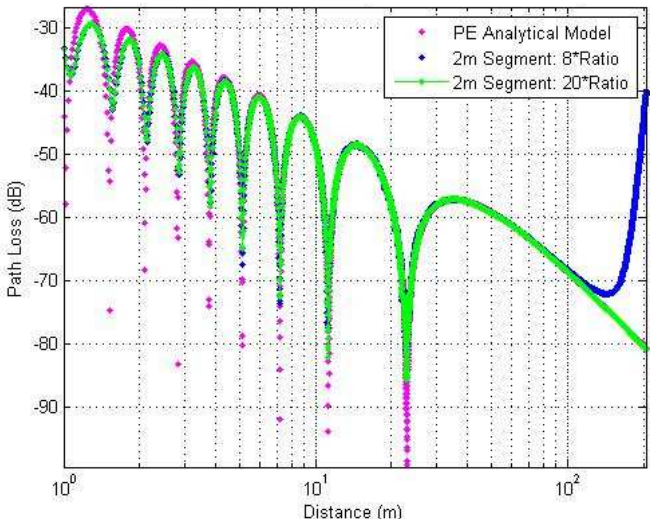

Fig. 8. 2m Segment stability vs. instability with different time schemes

Obviously, for a certain time scheme, an optimal (in terms of computational performance) segment size exists. The relationship between the total CPU execution time and the segment size can be calculated as:

$$Total\_CPU\_Time = n \cdot dt \cdot N\,, \qquad (4)$$

where n is the number of time steps (iterations) for each segment to reach its steady state, dt is the CPU time required for each single time step (iteration) in the segment and N is the number of segments divided from a problem. To maintain the stability of the SFDTD in the plane earth example we have discussed, Fig. 9 shows the total CPU time requirements.

VI. CONCLUSION

In conclusion, by reducing the segment size and taking the stability issue into consideration, the proposed SFDTD technique for implementing FDTD modeling allows the CPU execution time to increase only linearly with the number of segments instead of exponentially increasing as seen in a conventional FDTD.

The SFDTD method also enhances reusability of the simulation data. For example, we can use the saved interface array sources to further extend the simulation in terms of problem dimensions and changes to the simulated environment. However, note that the SFDTD method also requires that:

a. The PMLs are reasonably efficient as absorbing boundary conditions in order to reduce the overhead owing to each segment.

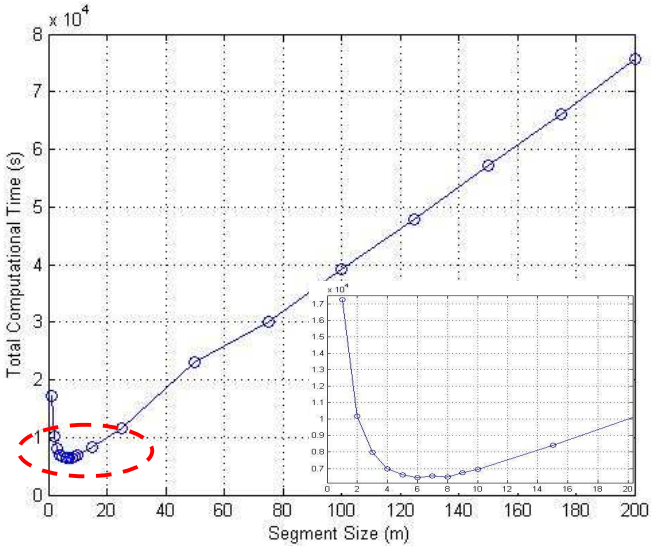

Fig. 9. Total computational time for the SFDTD stability in the plane earth model described in Section IV.

b. The size of the segments needs to be carefully chosen and should include features that have the potential to introduce significant effects on the equivalent source array. i.e., signal reflections from the next segment must be insignificant compared with the signal level in the current segment; otherwise, the equivalent array source loses its precision.

The 2D SFDTD technique will be applied to tunnels and to below to above ground situations (i.e., Fire Hydrants) in our future work.

REFERENCES

[1] K. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media", *IEEE Trans. Antennas Propag.*, vol. 14, pp. 302-307, 1966.

[2] Y. Wu, M. Lin and I. J. Wassell, "Path Loss Estimation in 3D Environments using a Modified 2D Finite-Difference Time-Domain Technique", The IET 7th International Conference on Computation in Electromagnetics, CEM 2008. pp. 98-99, Apr. 2008.

[3] D. M. Sullivan, *Electromagnetic Simulation using the FDTD Method*, Wiley-IEEE Press, pp. 8, Jul. 2000.

[4] M. Lin, Y. Wu, I. J. Wassell, "Wireless Sensor Network: Water Distribution Monitoring System", 2008 IEEE Radio and Wireless Symposium

[5] D. White, M. Stowell, J. Koning, R. Rieben, A. Fisher, N. Champagne, and N. Madsen, "Higher-Order Mixed Finite Element Methods for Time Domain Electromagnetics", Feb. 2004. http://www.llnl.gov/tid/lof/documents/pdf/304775.pdf

[6] Joseph W. Schuster, and Raymond J. Luebbers, "FDTD Techniques for Evaluating the Accuracy of Ray-Tracing Propagation Models for Microcells", Remcom, Inc. http://www.remcom.com/multimedia/publications/fdtd.pdf

[7] G. D. Kondylis, F. De Flaviis, G. J. Pottie and T. Itoh, "A Memory-Efficient Formulation of the Finite-Difference Time-Domain Method for the Solution of Maxwell Equations", *IEEE Trans. Microwave Theory Tech*, vol. 49, No. 7, pp. 1310-1320, Jul. 2001.

[8] W. Yu, R. Mittra, T. Su, and Y. Liu, "A Robust Parallelized Conformal Finite Difference Time Domain Field Solver Package Using the MPI Library", *IEEE Antennas and Propagation Magazine*, vol. 47, No. 3, pp. 354-360, Mar. 2005.

[9] A. Taflove and S. C. Hagness, *Computational Electrodynamics the Finite-Difference Time-Domain Method*, 3rd ed., Artech House Publishers, pp. 132, Jun. 2005.

[10] S. R. Saunders, Antennas and Propagation for Wireless Communication Systems, Wiley, Aug. 2004.