Graphics Processing Unit-Accelerated Implementation of the Plane Wave Time Domain Algorithm

Yang Liu [1], Vitaliy Lomakin [2], and Eric Michielssen [1]

[1] Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48105, USA
liuyangz@umich.edu, emichiel@umich.edu

[2] Department of Electrical and Computer Engineering
University of California, San Diego, CA 92093-0407, USA
vitaliy@ece.ucsd.edu

Abstract: A graphics processing unit (GPU) accelerated implementation of the multilevel plane wave time domain (PWTD) algorithm for rapidly evaluating transient electromagnetic fields generated by large-scale temporally bandlimited dipole constellations is presented. By porting the computationally most intensive stages of the multilevel PWTD algorithm onto a Tesla C2050 device, 20X speedups and 56% - 70% memory reduction are achieved over a single thread serial implementation on a Dual-Core AMD Opteron 2220 SE.

Keywords: plane wave time domain algorithm (PWTD), graphics processing unit (GPU), CUDA FORTRAN, CUFFT

1. Introduction

The multilevel plane wave time domain (PWTD) algorithm is a fast scheme that reduces the computational cost associated with the evaluation of transient electromagnetic fields produced by N_s dipoles active for N_t time steps from $O\left(N_t N_s^2\right)$ to $O\left(N_t N_s \log^2 N_s\right)$ [1-2], and hence can be used to accelerate the classical marching on in time scheme for solving time domain integral equations pertinent to the analysis of scattering from and radiation by complex and large-scale structures. Unfortunately, their favorably computational complexity notwithstanding, the computational costs of serial PWTD implementations still impede their applicability to real-life problems. Parallelization of the multilevel PWTD algorithm has been demonstrated on CPU clusters for analysis of EMC problems [3]. Recently, graphics processing units (GPU), viz. multi-threaded manycore processors that perform ultra-fast floating-point operations, have been shown to favorably compete with CPUs for parallelizing a variety of computational schemes to analyze electromagnetic phenomena [4-7]. The use of GPUs was facilitated by the introduction of the Compute Unified Device Architecture (CUDA), which provides a framework for writing general purpose GPU codes and offers a great flexibility in accelerating many commonly encountered scientific computing tasks. CUDA has been further extended to be incorporated into various programming environments resulting in powerful and convenient programing tools, such as CUDA FORTRAN. This paper describes the first GPU-accelerated implementation of the multilevel PWTD algorithm.

The paper is organized as follows. Formulations of direct and multilevel PWTD schemes for

evaluating transient electromagnetic fields from large-scale dipole constellations are briefly reviewed in Section 2. Their GPU implementations are described in Section 3. The performance of the GPU-accelerated PWTD implementation is discussed in Section 4. Conclusions of the paper are summarized in Section 5.

2. Formulations

A. Direct Scheme

Consider a current $\mathbf{J}(\mathbf{r},t)$ that consists of N_s dipoles

$$\mathbf{J}(\mathbf{r},t)=\sum_{n=1}^{N_s} f_n(t)\delta(\mathbf{r}-\mathbf{r}_n)\hat{\mathbf{u}}_n \tag{1}$$

where the nth dipole's temporal signature $f_n(t)$ is assumed bandlimited to ω_{max} and quasi time-limited to $0<t<T$. To numerically evaluate electric fields $\mathbf{E}(\mathbf{r},t)$ generated by this current, the time signature is discretized using time steps of size $\Delta_t=\pi/\beta\omega_{max}$ with a temporal oversampling factor $5<\beta<20$, and is represented by its $N_t=T/\Delta_t$ samples as

$$f_n(t)=\sum_{j=1}^{N_t} I_{n,j}T_j(t). \tag{2}$$

Here $I_{n,j}=f_n(j\Delta_t)$ and $T_j(t)=T(t-j\Delta_t)$ is a time-shifted local interpolant. For direct schemes, $T(t)$ is chosen to be a p_n^{th} order Lagrange interpolant that is time-limited to $-\Delta_t<t<p_n\Delta_t$. The electromagnetic field at $t_i=i\Delta_t$ along the mth dipole due to all other dipoles can be expressed as

$$\left\langle \hat{\mathbf{u}}_m,\mathbf{E}(\mathbf{r}_m,t_i)\right\rangle=-\frac{\mu_0}{4\pi}\sum_{n=1,n\neq m}^{N_s}\sum_{j=i-p_n-\lfloor|\mathbf{r}_m-\mathbf{r}_n|/c\Delta_t\rfloor}^{i-\lfloor|\mathbf{r}_m-\mathbf{r}_n|/c\Delta_t\rfloor} I_{n,j}\left[\hat{\mathbf{u}}_m\bullet\int_{-\infty}^{t}dt'\left(\partial_{t'}^2 I-c^2\nabla\nabla\right)\bullet\frac{T_{j-i}\left(t'-|\mathbf{r}-\mathbf{r}_n|/c\right)\hat{\mathbf{u}}_n}{|\mathbf{r}-\mathbf{r}_n|}\right]\Bigg|_{\substack{\mathbf{r}=\mathbf{r}_m\\ t=0}} \tag{3}$$

where $\langle\cdot,\cdot\rangle$ denotes the standard inner product and $\lfloor\cdot\rfloor$ is the floor truncation function. The cost of computing (3) for $m=1,2,...,N_s$ and $i=1,2,...,N_t$ scales as $O\left(N_tN_s^2\right)$.

A. Multilevel PWTD Scheme

The plane wave time domain (PWTD) algorithm permits computing interactions between dipoles in a group-wise manner, based on a transient plane wave decomposition of the radiated field. Assume that the mth and n^{th} dipoles reside in two non-overlapping spheres α and α' with radius R_s centered about $\mathbf{r}_\alpha^c$ and $\mathbf{r}_{\alpha'}^c$; let $R_{c,\alpha\alpha'}=\left|\mathbf{R}_{c,\alpha\alpha'}\right|=\left|\mathbf{r}_{\alpha'}^c-\mathbf{r}_\alpha^c\right|$. In the PWTD scheme, the temporal basis function is chosen as the approximate prolate spheroidal (APS) interpolant [8], which not only is band-limited to $\omega_s=\beta\omega_{max}$, but also approximately time limited to $-p_f\Delta_t<t<p_f\Delta_t$; assuming that the signal is bandlimited, the interpolation error associated with the use of this function decreases exponentially with increasing p_f. The time signature of the nth dipole is broken into N_l consecutive bandlimited subsignals

$$f_n(t)=\sum_{l}^{N_l} f_n^l(t)=\sum_{l}^{N_l}\sum_{j=(l-1)M_t+1}^{lM_t} I_{n,j}P_j(t) \tag{4}$$

where $P_j(t)=P(t-j\Delta_t)$ is the time-shifted APS interpolant and $N_lM_t=N_t$; M_t is chosen such that the duration of each subsignal $\left(M_t+2p_f\right)\Delta_t\le\left(R_{c,\alpha\alpha'}-2R_s\right)/c$. Let $\mathbf{E}_{n,l}(\mathbf{r},t)$ denote the electric fields in sphere α' generated by the lth subsignal of dipole n in sphere α. The field along dipole m can be expressed as

$$\left\langle \hat{\mathbf{u}}_m, \mathbf{E}_{n,l}\left(\mathbf{r}_m, t\right)\right\rangle = \frac{\eta_0}{8\pi^2 c^2}\sum_{p=0}^{K}\sum_{q=-K}^{K} w_{pq}\left[S^{-}_{m,\alpha'}\left(\hat{\mathbf{k}}_{pq}, t, \hat{\mathbf{u}}_m\right)\right]^T \underset{3}{*}\mathcal{T}\left(\hat{\mathbf{k}}_{pq}, t, K\right)\underset{2}{*}\left[S^{+}_{n,\alpha}\left(\hat{\mathbf{k}}_{pq}, t, \hat{\mathbf{u}}_n\right)\right]\underset{1}{*} f_n^l\left(t\right) \tag{5}$$

where w_{pq} and $\hat{\mathbf{k}}_{pq}$ are quadrature weights and points for integration on a unit sphere, $\underset{i}{*}$, $i=\{1,2,3\}$ denotes standard convolution, and the translator is

$$\mathcal{T}\left(\hat{\mathbf{k}},t,K\right) = \frac{c\partial_t^2}{2R_{c,\alpha\alpha'}}\sum_{\upsilon=0}^{K}\left(2\upsilon+1\right)P_\upsilon\left(ct/R_{c,\alpha\alpha'}\right)P_\upsilon\left(\hat{\mathbf{k}}\cdot\mathbf{R}_{c,\alpha\alpha'}/R_{c,\alpha\alpha'}\right), \quad \text{for } |t| \le R_{c,\alpha\alpha'}/c\,. \tag{6}$$

Here $P_\upsilon\left(\cdot\right)$ is the Legendre polynomial of degree υ, $K=\lfloor 2\chi\omega_s R_s/c\rfloor+1$ with an oversampling factor χ, and

$$S^{\pm}_{o,g}\left(\hat{\mathbf{k}}, t, \mathbf{v}\right) = \hat{\mathbf{k}}\times\mathbf{v}\,\delta\left(t \pm \hat{\mathbf{k}}\bullet\left(\mathbf{r}_o - \mathbf{r}_g^c\right)/c\right) \tag{7}$$

where $o=\{m,n\}$ and $g=\{\alpha',\alpha\}$. Equation (5) indicates that interactions between two sets of well-separated dipoles can be evaluated by a three stage scheme: 1. "Construct outgoing rays" by performing convolution $\underset{1}{*}$ for all dipoles in the source sphere. 2. "Translate outgoing rays" to incoming rays in the observation sphere by performing convolution $\underset{2}{*}$. 3. "Project incoming rays" along dipoles in the observation sphere by performing convolution $\underset{3}{*}$ and the leftmost summation.

In the multilevel PWTD scheme, the computational domain is divided recursively into N_v levels [1-2]. Assume that the dipoles are located on a surface, there are $4^{N_v - v}$ groups at level v, $v=1,2,\ldots,N_v$ and each group interacts with $O(1)$ other groups by PWTD. For level $v>1$, the outgoing rays are constructed from level $v-1$ outgoing rays by spherical interpolation. Similarly, the incoming rays at level $v>1$ are added onto level $v-1$ incoming rays following spherical filtering. The interactions between group pairs that haven't been accounted for by PWTD are evaluated by the direct scheme; this computation is referred to as "near-field calculation"; all other PWTD computations are termed : "far-field". Upon fixing the size of level 1 groups to a fraction of the wavelength at frequency ω_{max}, the computational complexity of the multilevel PWTD algorithm scales as $O\left(N_t N_s \log^2 N_s\right)$.

3. GPU Implementations

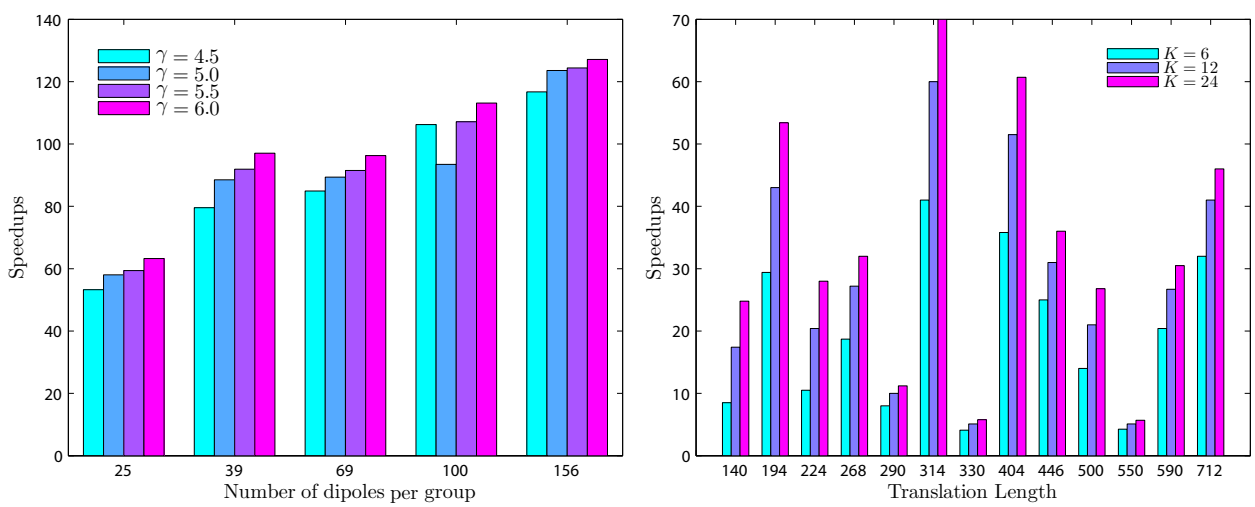
To accelerate the multilevel PWTD scheme utilizing the computational power of a GPU, two of the computationally most intensive PWTD stages, viz. near-field calculation and field translation, are implemented using the CUDA FORTRAN programming interface on a Tesla C2050 device with 515Gflops of double precision floating point performance and 3GB of memory. Other PWTD stages are run serially on an AMD Opteron 2220 SE at 2.79 GHz with 16GB of RAM.

A. Near-field Calculation

As discussed earlier, all PWTD near-field contributions are calculated by the direct scheme. To efficiently utilize the limited GPU memory, all near-field interpolation coefficients are calculated on-the-fly on the GPU while coordinates and directions of dipoles, sources and fields, as well as near-field interaction lists (NILs) are dynamically allocated in the global memory at every time step i. In GPU vernacular, this parallelization strategy is best described as "one block per group" and "one thread per observer". Each block loops over the NIL of the corresponding group and updates fields due to related source groups. Shared memory is loaded with dipole coordinates and directions for each nearfield group pair. Fig. 1(a) shows speedups for the near-field calculation stage. Increasing the number of dipoles per group or the separation cutoff (equivalently the size of NIL) results in higher speedups due to larger number of threads/blocks and larger computational load for each thread.

B. Translation

The translation stage is the computationally most intensive one of the multilevel PWTD algorithm and its GPU implementation plays a critical role in overall code performance. As mentioned earlier, translation is executed level by level. Interactions between each far-field group pair at level v is calculated by (5). Convolution of outgoing rays with translators consists of three operations, viz. forward fast Fourier transformation (FFT) of outgoing rays, multiplication of outgoing rays by translators in the frequency domain, and backward FFTin the result to incoming rays. These operations are implemented on a GPU as follows: at $i = lM_t(v)$, the outgoing rays, incoming rays, and translators at level v are transferred into global memory. The CPU then loops over far-field pairs associated with the same translator and GPU kernels are launched for each group pair. Multiplying frequency domain translators by outgoing rays can be readily parallelized using a "one thread per data point" strategy while the forward and inverse FFT operations are implemented using the CUFFT library [9]. As a result of this strategy, $(K+1)(2K+1)$ transforms are executed simultaneously on the GPU. Fig. 1(b) shows speedups for translation between one far-field pair with varying K and translation length (far-field pair distance). Compared to FFTW-based CPU translations, larger K leads to higher speedups because larger batches of FFTs are launched; however, larger transform length will not always increase the speedup because of implementation differences between the CUFFT and FFTW libraries.



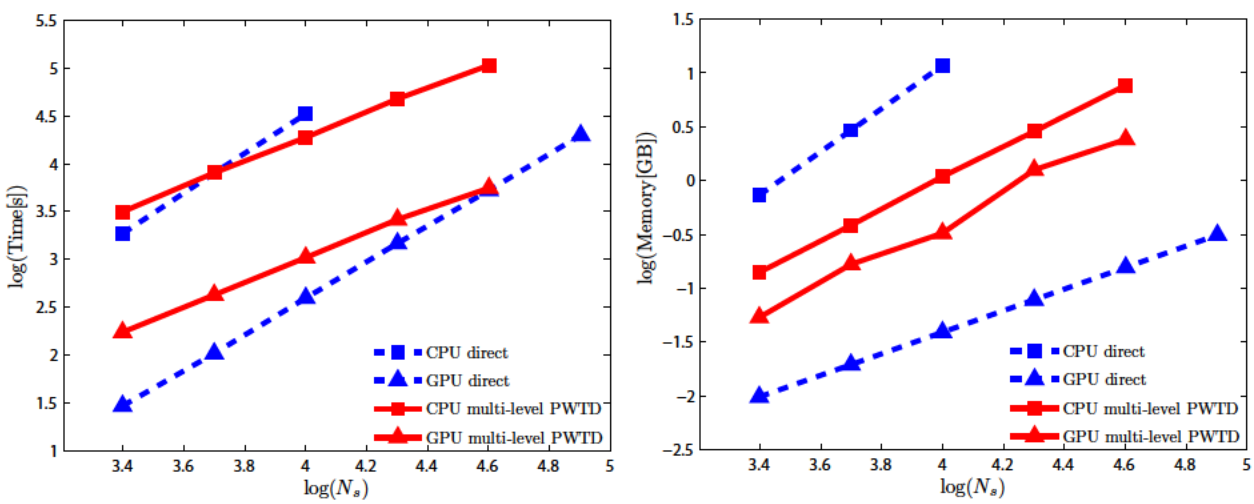
(a) (b)

Fig. 1. Speedups of GPU (Tesla C2050) implementations over CPU (AMD Opteron 2220 SE) implementations for (a) Near-field calculation stage, (b) Translation between one far-field group pair.

4. Overall Performance

The GPU-accelerated implementations of the direct and the multilevel PWTD schemes are tested by the following example. A large number of randomly oriented dipoles are distributed uniformly on a square plate with side length varying from 1.5m $(N_s = 2{,}500)$ to 8.5m $(N_s = 80{,}000)$. The time signature of each dipole is bandlimited to $f_{max} = 1\text{GHz}$ and given by $f_n(t) = e^{-(t-6\sigma)^2/2\sigma^2}$ with $\sigma = 0.955\text{ns}$. The fields are computed for $N_t = 500$ time steps with $\Delta_t = 6.25\times10^{-2}\text{ns}$. The L^2 norm error between the PWTD results and the exact results is controlled to 1×10^{-4}. Fig. 2(a) shows the computational time of the direct scheme and the multilevel PWTD scheme for CPU (AMD Opteron 2220 SE) implementations and GPU (Tesla C2050) accelerated implementations. The speedups are 70X – 90X for the direct scheme and 20X for the multilevel PWTD scheme. Because of the lower speedups of the

multilevel PWTD scheme compared to the direct one, the cross-over point, viz. the dipole count where the PWTD scheme becomes faster than its direct counterpart shifts from $N_s = 5{,}000$ for the CPU implementation to $N_s = 40{,}000$ for the GPU implementation. However, dominated by the scaling laws for the computational complexity, the GPU-accelerated multilevel PWTD implementation will perform better than the direct scheme as N_s increases. The memory required by the CPU and the GPU is plotted in Fig. 2(b). For the direct scheme, the GPU-accelerated implementation only requires $O(N_s)$ global memory to store dipole locations, directions, source magnitude, fields and NILs, as opposed to the additional $O(N_s^2)$ requirement to store interpolation coefficients for the CPU implementation. For the multilevel PWTD scheme, the GPU only requires storage of outgoing rays, incoming rays, and translators at one level, resulting in 56% - 70% memory reduction from that used by the CPU implementation. Limited by the total of 3GB global memory on Tesla C2050, the maximum problem size of a pure GPU-accelerated multilevel PWTD implementation is $N_s = 40{,}000$ with $N_s N_t = 2 \times 10^7$ data points.



(a) (b)

Fig. 2. (a) Comparison of the computational time of the direct and the multilevel PWTD schemes for both serial CPU (AMD Opteron 2220 SE) implementations and GPU (Tesla C2050) accelerated implementations. (b) Comparison of the memory required by CPU and GPU for the direct and the multilevel PWTD schemes.

5. Conclusions

This paper presented a GPU-accelerated implementation of the multilevel PWTD algorithm. With the near-field calculation and translation stages accelerated by the GPU, our implementations exhibit a 20X speedup and 56% - 70% memory reduction over a serial CPU implementation. Current research focuses on multi-GPU-accelerated implementations of the PWTD algorithm.

References

[1] A. A. Ergin, B. Shanker, and E. Michielssen, "The plane-wave time-domain algorithm for the fast analysis of transient wave phenomena," *Antennas and Propagation Magazine, IEEE,* vol. 41, pp. 39-52, 1999.

[2] B. Shanker, A. A. Ergin, M. Lu, and E. Michielssen, "Fast analysis of transient electromagnetic scattering phenomena using the multilevel plane wave time domain algorithm," *Antennas and Propagation, IEEE Transactions on,* vol. 51, pp. 628-641, 2003.

[3] N. Liu, M. Lu, B. Shanker, and E. Michielssen, "The parallel plane wave time domain algorithm-accelerated marching on in time solvers for large-scale electromagnetic scattering

problems," in *Antennas and Propagation Society International Symposium*, 2004, pp. 4212-4215 Vol. 4.

[4] N. A. Gumerov and R. Duraiswami, "Fast multipole methods on graphics processors," *Journal of Computational Physics*, vol. 227, pp. 8290-8313, 2008.

[5] M. Cwikla, J. Aronsson, and V. Okhmatovski, "Low-frequency MLFMA on graphics processors," *Antennas and Wireless Propagation Letters, IEEE,* vol. 9, pp. 8-11, 2010.

[6] S. Li, B. Livshitz, and V. Lomakin, "Fast evaluation of Helmholtz potential on graphics processing units (GPUs)," *Journal of Computational Physics*, 2010.

[7] S. Li, R. Chang, and V. Lomakin, "Fast Electromagnetic Integral Equation Solvers on Graphics Processing Units," *GPU Computing Gems Jade Edition*, p. 243, 2011.

[8] J. Knab, "Interpolation of band-limited functions using the approximate prolate series (Corresp.)," *Information Theory, IEEE Transactions on*, vol. 25, pp. 717-720, 1979.

[9] C. NVIDIA, "CUFFT library 4.0," ed: NVIDIA, 2011.