POLITECNICO DI TORINO
Repository ISTITUZIONALE

FFT-based solvers for the EFIE on graphics processors

(Article begins on next page)

19 April 2024

FFT-based solvers for the EFIE on Graphics Processors

*M. A. Francavilla[1], F. Vipiana[2] and G. Vecchi[1]
[1]Antenna and EMC Lab (LACE), Electronics Dep., Politecnico di Torino
C.so Duca degli Abruzzi 24, 10129 Torino, Italy
[2]Antenna and EMC Lab (LACE), Istituto Superiore Mario Boella (ISMB)
Via Pier Carlo Boggio 61, Torino 10138, Italy
Emails: matteo.francavilla@polito.it, vipiana@ismb.it, giuseppe.vecchi@polito.it

Introduction

Recently Graphic Processing Units (GPUs) have reached very high computing and memory capabilities; at the same time the price for these devices has dropped, making them suitable for standard workstations. GPUs of the current generation are able to perform vector operations at much higher rates with respect to Central Processing Units (CPUs). The reason for this discrepancy in floating-point capability lies in the specialization of GPUs for compute-intensive highly parallel computation, exactly what graphics rendering is about.

Due to the high capabilities of these devices, at a relatively cheap price, many numerical applications can be sped up by moving part of the computations from the CPU to the GPU. At the aim of simplifying this task, NVIDIA introduced a parallel computing architecture named "Compute Unified Device Architecture" (CUDA) [1], which provides instruments to the programmers to program the GPU with a low effort.

In computational electromagnetics the problems often resort to the solution of a linear system of equations, with a number of unknowns that can be in the order of hundreds of thousands for usual scattering and antenna problems. The solution of linear systems can be efficiently parallelized, and the GPUs allow for a parallelization of the computations on a standard workstation. Here we propose a GPU-acceleration of a fast solver for the Electric Field Integral Equation (EFIE) discretized through the Method of Moments (MoM).

Formulation of the problem

The integral equation for Perfectly Electric Conductors (PECs) in free space is discretized by the MoM with the usual Rao-Wilton-Glisson basis [2]. The computation of the system matrix can be parallelized and accelerated onto GPUs, as shown in [3]. Here we focus our attention on the solution of the linear system when employing a fast solver. In [4] the Fast Multipole Method (FMM) acceleration is described. Here instead we refer to a fast formulation, based on the

Green's function interpolation accelerated by the Fast Fourier Transform (FFT), as presented in [5][6] for planar arrays, and in [7] for 3D PEC objects.

This formulation can be classified into the grid-based approaches, where the free space Green's function is mapped onto a regular Cartesian grid. Here the Green's function is expanded in terms of Lagrange polynomials, defined over a regular grid. The system matrix is then factorized into a product of sparse matrices. As common for all fast IE solvers, a sparse matrix containing the strong interactions must be added, due to the poor approximation of the Green's function near its singularity.

The linear system is solved through an iterative solver, based on the BiConjugate Gradient Stabilized (BiCGStab) method [8], which requires the evaluation of several sparse products at each iteration. Grid-based methods allow for an efficient evaluation of the sparse products through the Fast Fourier Transform (FFT) algorithm, thanks to the circulant properties that can be obtained after some algebraic manipulations.

GPU implementation

The main aim of our work is implementing the sparse products on the GPU. A limit of GPU computation at present is given by the memory bandwidth of the GPU itself; memory transfers from CPU to GPU and vice versa are usually the bottleneck for applications, and efficient implementations can be obtained when data transfers are minimized. Therefore at each iteration only the right hand side vector must be loaded onto the device, while the matrices are transferred to the GPU memory at once.

For sparse matrix-vector multiplication several algorithms are known in literature, the efficiency strongly varying based on the chosen storage scheme and the sparsity pattern of the matrix. Here we implement the algorithm described in [9].

Moreover we use a very efficient implementation of the FFT algorithm directly provided by NVIDIA through the CUFFT library [10].

Numerical results

To verify the acceleration obtained by using the GPU in cooperation with the CPU, we conducted some experiments comparing the time per iteration obtained with this approach, with respect to the same solver ran on one CPU. The numerical simulations were performed on an Intel Xeon E5440 (2.83 GHz) 64-bit workstation with 32GB of RAM memory. The graphic device is a NVIDIA Tesla C1060 (240 cores, 4GB of memory).

The test case is a sphere discretized with triangles of edge around $\lambda/10$ at the frequency of 1 GHz. The radius of the sphere is increased from λ to 4.2λ, keeping the mesh density constant. The iteration times are compared in Figure 1, and the speed-up factor of the GPU approach is around 6 for the scenario with about 80,000 unknowns.
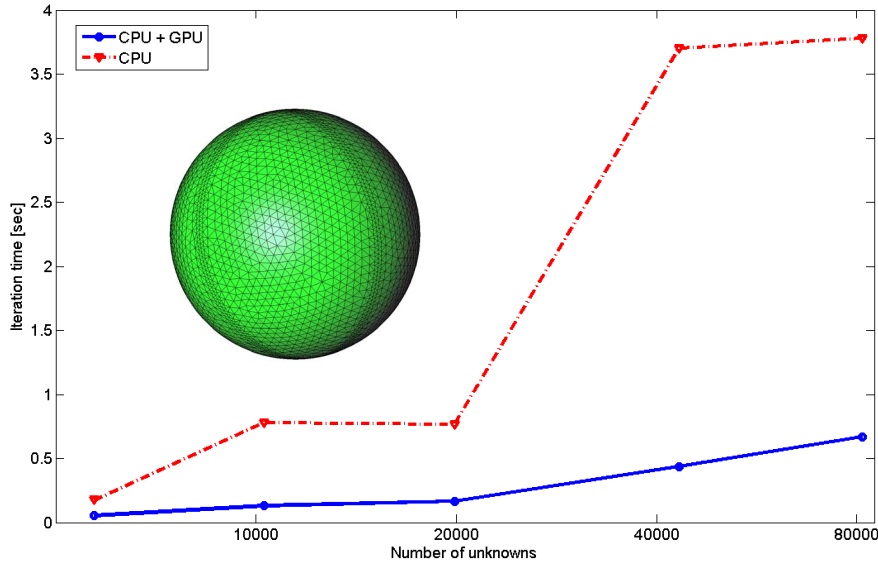

Figure 1: Time per iteration of the BiCGStab solver. Inset: test case with radius 0.42m, discretized with N = 10,522 unknowns.

The CPU version of the solver uses the FFTW implementation for the FFT algorithms, which is the most efficient serial implementation of the algorithm at present. It is worth noting that the FFTW algorithms adapt very well with arbitrary dimensions of the matrices to transform, and almost piece-wise constant behaviors like the one shown are not unexpected. On the other hand we verified that CUFFT performs very well only when the dimensions are powers of 2, and it is convenient to adopt zero padding strategies, provided that memory is not an issue.

Acknowledgment

References

[1] NVIDIA CUDA: Compute Unified Device Architecture Programming Guide, Version 2.3, NVIDIA Corp., January 2009.

[2] S. M. Rao, D. R. Wilton, A. W. Glisson, "Electromagnetic scattering by surfaces of arbitrary shape," IEEE Trans. Antennas Propagation, vol. 30, pp. 409–418, May 1982.

[3] S. Peng and Z. Nie, "Acceleration of the method of moments calculations by using Graphic Processing Units", IEEE Transactions on Antennas and Propagation, vol. 56, No. 7, July 2008, pp. 2130-2133.

[4] N. A. Gumerov and R. Duraiswami, "Fast multipole methods on graphic processors", Journal of Computational Physics, vol. 227, No. 18, September 2008, pp. 8290-8313.

[5] B. J. Fasenfest, F. Capolino, D. R. Wilton, D. R. Jackson, and N. J. Champagne, "A fast MoM solution for large arryas: Green's function interpolation with FFT," IEEE Antennas Wireless Propagation Letters, vol. 3, pp. 161–164, 2004.

[6] B. J. Fasenfest, F. Capolino, and D. R. Wilton, "Preconditioned gifft: A fast mom solver for large arrays of printed antennas," ACES J., vol. 21, pp. 276–283, Nov. 2006.

[7] S. Seo and J. Lee, "A fast IE-FFT algorithm for solving PEC scattering problems," IEEE Trans. Magn., vol. 41, pp. 1476–1479, May 2005.

[8] H. A. Van der Vorst, "Bi-CGStab: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 13, pp. 631–644, 1992.

[9] Nathan Bell and Michael Garland, "Efficient Sparse Matrix-Vector Multiplication on CUDA", "NVIDIA Technical Report NVR-2008-004", December 2008.

[10]   CUFFT Library, Version 2.3, NVIDIA Corp., June 2009.

[11]   http://www.fftw.org