Speedup of Integral-FEM Based Solution of Eddy Currents Testing Problems Using Graphical Processing Unit

Gabriel Preda, Valentin Popa, and Florea I. Hantila
Politehnica University of Bucharest, Electrical Engineering Department
Splaiul Independentei 313, Bucharest, Romania, 064213, Romania
gabriel.preda@upb.ro

Abstract — We will show how to speedup the solution of an integral-FEM formulated problem using graphical processing units (GPU). With this method, both the phase of forming the problem system matrix and the solution of this system are considerably accelerated.

I. INTRODUCTION

Recent advances in computational electromagnetics shows a trend toward increased problem complexity, paired by a challenge to maintain scalability of the implemented solutions, to take advantage of the increased calculation power made available by CPU manufacturers. Highly scalable parallel solutions using MPI on multi-core systems or on clusters of multi-core machines were implemented in past years and impressive progresses was made in the field of high performance computing techniques applied to electromagnetic simulations. Exciting opportunities toward further increase of computational performance are offered the use of Graphical Processing Units (GPU). Each GPU node in such a system comprise from several tens to several hundreds of independent streams processors.
One single GPU may be the equivalent of a medium-size CPU cluster computer. The software development environment that allows parallel computation on the GPU is called compute unified device architecture (CUDA).

Several applications of Graphical Processing Units based implementation in computational electromagnetics were communicated over last year, including a study on sparse vector-matrix multiplication, with a super-linear acceleration factor [1], solutions for micromagnetic problems calculations [2][3], different implementations for wave propagation problems [4][5] or parallel simulation method based on the radial point interpolation method [6].

Electromagnetic nondestructive testing simulations require intensive calculation resources, successive solutions of forward problems being performed to solve an inverse problem, typically an estimation of a defect geometry from a measured signal. For such problems, parallel solutions being an option for accelerating the computation. An integral formulation for pulse eddy currents, using MPI parallelization, was the subject of the a previous paper [7]. In the current paper we demonstrates the efficiency of a GPU based-implementation for speedup the calculation of system matrix terms and the factorization of the system matrix for the same problem.

II. FORMULATION

The formulation presented here is based on application of **T**- electric vector potential to the integral equation of eddy currents, like in [7]. Starting from Maxwell equations in quasi-stationary form and the constitutive relationship:

$$\boldsymbol{E} = \rho \cdot \boldsymbol{J}\ , \tag{1}$$

where $\boldsymbol{J}$ is the current density, $\boldsymbol{E}$ is the electrical field and ρ is the resistivity in the conductive domain Ω_c. In the specimen coordinates frame, the electrical field is:

$$\mathbf{E} = -\frac{\partial \mathbf{A}}{\partial t} - \nabla V\ , \tag{2}$$

where V is the electric scalar potential and $\boldsymbol{A}$ is magnetic vector potential. The magnetic vector potential can be calculated using Biot-Savart formula:

$$\mathbf{A} = \frac{\mu_0}{4\pi}\int_{\Omega}\frac{\mathbf{J}}{r}\mathrm{d}v + \mathbf{A}_0\ , \tag{3}$$

with $\mathbf{A}_0$ being the magnetic vector potential due to the impressed current sources:

$$\mathbf{A}_0 = \frac{\mu_0}{4\pi}\int_{\Omega_0}\frac{\mathbf{J}_0}{r}\mathrm{d}v \tag{4}$$

and Ω_0 being the air. Only conductive media are meshed. The current density is expressed in terms of shape functions associated to the edges in the inner co-tree:

$$\mathbf{J} = \sum_{k=1}^{n}\mathbf{N}_k \nabla\times\mathbf{T}_k\ . \tag{5}$$

Applying Galerkin approach, the following equation system is obtained:

$$[R]\{I\} + [L]\frac{d\{I\}}{dt} = \{U\}\ , \tag{6}$$

where the terms of matrices [$\boldsymbol{R}$] and [$\boldsymbol{L}$] are:

$$L_{ij} = \frac{\mu_0}{4\pi}\int_{\Omega_c}\int_{\Omega_c}\frac{\nabla\times\mathbf{N}_i\cdot\nabla\times\mathbf{N}_j}{r}dv_c dv_c\ , \tag{7}$$

$$R_{ij} = \int_{\Omega_c}\nabla\times\mathbf{N}_i\cdot\rho\nabla\times\mathbf{N}_j dv_c\ , \tag{8}$$

The *[L]* matrix is full, having all terms calculated as sums over tethraedras of double volume integrals from the kernel 1/*r*:

$$L_{ij} = \frac{\mu_0}{4\pi} \sum_{\omega_k \supset i} \nabla \times \mathbf{N}_i \sum_{\omega_l \supset j} \nabla \times \mathbf{N}_j \int_{\omega_k} \int_{\omega_l} \frac{1}{r_{kl}} dv dv \quad (9)$$

Computation of terms of the matrix *[L]*is extremely CPU intensive and will be subjected to graphical processing unit based parallelism. The solution of the forward problem modeled using the above presented formulation is used to build a database of signal-defect pairs. A set of such signal-defect pairs is propagated then through a neural network for training.

III. PARALLEL ALGORITHM IMPLEMENTATION

The critical computational parts are two: calculation of inductance matrix *[L]* and *LU* decomposition step in the solution of the linear equation system. Whilst the former is intrinsically parallel and requires virtually no synchronization between the nodes, the later requires adaptation of the algorithm in order to use the full potential offered by the special architecture of GPU.

For the first problem, the challenge is to select the best partitioning of the system matrix, in order to minimize the memory requirement for each CUDA thread.

We will focus on the computation of the terms of the double volume integral:

$$\tau_{kl} = \int_{\omega_k} \int_{\omega_l} \frac{1}{r_{kl}} dv dv \quad (10)$$

where:

$$L_{ij} = \frac{\mu_0}{4\pi} \sum_{\omega_k \supset i} \nabla \times \mathbf{N}_i \sum_{\omega_l \supset j} \nabla \times \mathbf{N}_j \cdot \tau_{kl} \quad (11)$$

We split the matrix with terms τ_{kl} into small size blocks. A pair of block information is flattened in a data vector and is passed to the GPU device to compute the kernel terms for the pair of blocks. After calculation, the blocks of τ_{kl} terms are assembled the full matrix *[L]* on the host system is calculated using CPU with relative small computational effort.

For the *LU* decomposition, the second inner loop is subjected to parallel computation using CUDA.

IV. RESULTS AND CONCLUSIONS

The simulation setup for the test problem consists in a conductive plate, a pancake coil used to energize the specimen and a Hall sensor to pick-up the signal, like in [7]. The plate is 16 cm × 16 cm, with 10 mm thickness and having conductivity $\sigma = 10^6$ S/m. Coil dimensions are inner radius $R_{min} = 2$ mm, outer radius $R_{max} = 5$ mm, axial length $l_z = 4$ mm, liftoff $z = 0.4$ mm. The pickup sensor measures the magnetic flux density and is placed in the coil axis, at $z = 0.4$ mm. The coil signal used is a 70 μs, trapezoidal shaped pulse, and with additional rise and fall intervals of 10 μs each, with amplitude $I_{max} = 2000$ AT, and with a repetition frequency of 50 Hz. 55 time steps are simulated for a single pulse.

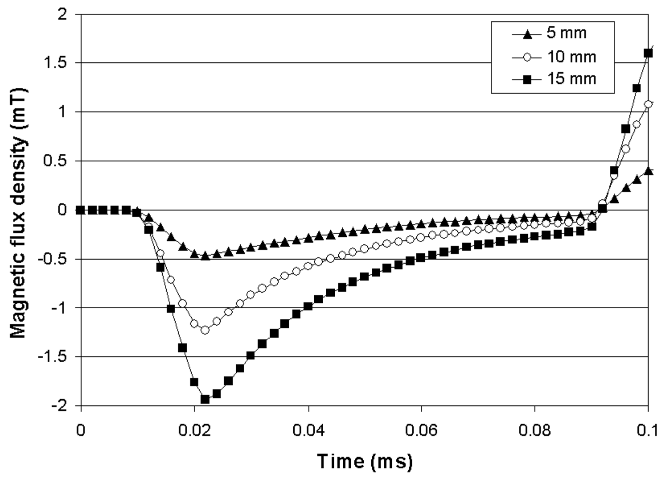

Fig. 1. Difference signal (magnetic flux density – z component) versus time; outer defects OD 80%, with 5, 10 and 15 mm length; scan point is y=0.0 mm.

The system used in the first experiments has a 32 bit, Core (2) Quad CPU, 2.6 GHz, with 3GB RAM and a NVIDIA *GeForce* 8600 GTS graphical card, with 4 nodes and 32 threads. For a problem discretized with 3600 tetra elements in the conductive domain, calculation of *[τ]* matrix terms was 56 sec. with CPU. For the *LU* decomposition the time with CPU was 62 sec. We can provide only preliminary results for GPU calculation time, of *LU* decomposition, for the moment. The best time obtained until now, with incomplete parallelism exploited, was 29 sec. for the problem size mentioned above. GPU times include computation time and memory transfer time. In the full paper we will present also the results for the inverse problem solution and we will develop further the parallel algorithms using CUDA.

V. REFERENCES

[1] D. M. Fernández, D. Giannacopoulos and W. J. Gross “Efficient Multicore Sparse Matrix-VectorMultiplication for FE Electromagnetics”, *IEEE Trans. on Magn.*, vol.45, no.3, pp. 1392-1395, 2009.

[2] S. Li, B. Livshitz and V. Lomakin, “Graphics processing unit accelerated *O(N)* micromagnetic solver”, *IEEE Trans. on Magn.*, vol.46, no.6, pp. 2373-2375, 2010.

[3] A. Kákay, E. Westphal and R. Hertel, “Speedup of FEM micromagnetic simulations with graphical processing units”, *IEEE Trans. on Magn.*, vol.46, no.6, pp. 2303-2306, 2010.

[4] N. Gödel, N. Nunn, T. Warburton and M. Clemens, “Scalability of higher-order discontinuous Galerkin FEM computations for solving electromagnetic wave propagation problems on GPU clusters”, *IEEE Trans. on Magn.*, vol.46, no.6, pp. 3469-3472, 2010.

[5] P. Sypek, A. Dziekonski and M. Mrozowski “How to render FDTD computations more effective using a graphics accelerator”, *IEEE Trans. on Magn.*, vol.45, no.3, pp. 3469-3472, 2009.

[6] S. Nakata, Y. Takeda, N. Fujita and S. Ikuno, “Parallel algorithm for meshfree radial point interpolation method on Graphics Hardware”, *IEEE Trans. on Magn.*, to be published.

[7] G. Preda, M. Rebican and F.I. Hantila, “Integral formulation and genetic algorithms for defects geometry reconstruction using pulse eddy currents”, *IEEE Trans. on Magn.*, vol. 46, no. 8, pp. 3433-3436, 2010.