CMRR Report Number 34, Summer 2010

FASTMAG: FAST MICROMAGNETIC SOLVER FOR LARGE-SCALE SIMULATIONS

R. Chang, S. Li, M.V. Lubarda, B. Livshitz, and V. Lomakin
University of California, San Diego

1 Introduction

Micromagnetic solvers have a high predictive power and are important for our ability to analyze and design magnetic components. Simulating a complex structure may be very time-consuming and the development of fast computational methods for micromagnetics is of high importance. Modern and future computational tools should rely on parallelization to allow for continuing scaling of the computational power. Conventionally, micromagnetic solvers have been parallelized on shared memory computers or CPU clusters but such implementations have limitations. Shared memory computers are limited by a relatively small number of available cores. Large clusters are expensive, consume much power, are available only as specialized facilities, and may often suffer from communication speed limitations. New massively parallel Graphics Processing Unit (GPU) computer architectures have emerged, offering massive parallelization at a very low cost.

Earlier our group has demonstrated efficient GPU implementations of finite difference micromagnetic solvers [1]. Here, we present a new fast micromagnetic solver, referred to as FastMag, which can handle problems of a small or very large size with a high speed. The method discretizes the computational domain into tetrahedral elements so that it is highly flexible for any geometries. FastMag allows handling any uniform or non-uniform geometries, does not require solving a linear system of equations, and requires very little memory. The results demonstrate a high efficiency and flexibility of the code. FastMag and its extensions can be used to micromagnetically model general magnetic structures, such as, a uniform and non-uniform arrays of generally shapes magnetic dots, magnetic wires, recording heads, magnetic media.

2 Problem formulation

Micromagnetic phenomena are governed by the Landau-Lifshitz-Gilbert (LLG) equation, which can be written in the following normalized form

$$\frac{\partial \mathbf{m}}{\partial t} = \frac{-\gamma}{1+\alpha^2}\left[\mathbf{m}\times\mathbf{H}_{\text{eff}} + \alpha\mathbf{m}\times(\mathbf{m}\times\mathbf{H}_{\text{eff}})\right], \quad (1)$$

where $\mathbf{m} = \mathbf{M}/M_s$ is the magnetization unit vector with the saturation magnetization M_s, t is time, y is the gyromagnetic ratio, and α is the damping constant.

The effective magnetic field $\mathbf{H}_{eff}$ in Eq. (1) is comprised of the external field $\mathbf{H}_{ext}$, anisotropy field $\mathbf{H}_{ani}$, exchange field $\mathbf{H}_{exc}$, and magnetostatic field $\mathbf{H}_{ms}$:

$$\begin{aligned}
&\mathbf{H}_{\text{eff}} = \mathbf{H}_{\text{ext}} + \mathbf{H}_{\text{ani}} + \mathbf{H}_{\text{exc}} + \mathbf{H}_{\text{ms}} \\
&\mathbf{H}_{ani} = (\mathbf{k}\cdot\mathbf{m})\mathbf{k} \\
&\mathbf{H}_{exc} = (2M_s l_{ex}^2 / H_K)\nabla^2\mathbf{m} \\
&\boldsymbol{H}_{ms} = \frac{M_S}{H_K}\left(-\nabla\iiint_V \frac{\nabla'\cdot\mathbf{m}}{|\mathbf{r}-\mathbf{r}'|}dV' + \nabla\oiint_S \frac{\mathbf{m}\cdot\mathbf{n}'}{|\mathbf{r}-\mathbf{r}'|}dS'\right).
\end{aligned} \quad (2)$$

Here, the anisotropy field is assumed to be uniaxial, $l_{ex}=A^{1/2}/M_s$ is the exchange length with the exchange constant A. The external field is a prescribed function of space and time. The magnetostatic field is given as a volume integral over the effective volume charges $\nabla' \cdot \mathbf{m}$ and surface integral over the effective surface charges $-\mathbf{m}\cdot\mathbf{n}'$ defined with respect to the normal to the computational domain surface $\mathbf{n}'$.

The magnetic structure of interest is discretized into a mesh of tetrahedrons and standard scalar linear basis functions are used to expand the magnetization. Let the total number of tetrahedral elements be N_T, the total number of nodes (i.e. tetrahedron's vertices) be N_V, and the number of surface nodes be N_s . Assume that the connectivity matrix $e(i)$ is available that defines the elements e in terms of nodes i. Assume also that a connectivity matrix $i(e)$ is available that relates the nodes to the elements containing the nodes. Such matrices can be given by meshing software. The magnetization is given at the nodes and it can be represented in space as

$$\mathbf{m} \cong \sum_{e'=1}^{N_T}\sum_{i'=1}^{4}\mathbf{m}_{e'(i')}N_{e'(i')} = \sum_{j=1}^{N_V}\mathbf{m}_j\varphi_j \quad . (3)$$

A discrete form of LLG is obtained by substituting the expansion in Eq. (3) into Eq. (1) and computing the resulting effective field at the nodes. A critical component of solving the LLG equation is the evaluation of the effective field.

3 Calculations of the effective field

3.1 Local fields: Anisotropy, external, and exchange

Computing the external and anisotropy fields is straightforward. These fields are directly sampled at the nodes. The exchange field is decomposed into its three Cartesian components, tested by the same functions as the basis functions, and averaged from the elements to nodes via the box method [2].

3.2 Magnetostatic field

The magnetostatic field is evaluated via superposition in a four-step procedure. First, equivalent magnetic charge densities are computed at the elements and nodes. Second, magnetic potentials at N_V nodes from collocated N_V source points are expressed as a dense matrix-vector product (MVP) problem, which is sparsified and evaluated via NGIM [1]. Third, the nodal potentials are corrected with analytical calculations of near-field interactions. Fourth, the magnetostatic field is obtained as the gradient of the scalar potential.

All integrals are evaluated numerically using a quadrature rule and singularity extraction procedure similar to the approaches used in the framework on electromagnetic integral equations [3]. We choose a 4-point rule in which the quadrature points coincide with the nodes (vertices) defining the elements, which results in a high accuracy and in a very small number of operations compared to any conventional quadrature rules with nodes defined inside the elements. The entire procedure can be summarized in the following matrix form

$$\boxed{\mathbf{H}_{ms} = [\mathbf{Z}][\mathbf{m}] \cong [\mathbf{Q}]^T \left([\mathbf{Z}_l] + [\mathbf{Z}_0]\right)[\mathbf{Q}][\mathbf{m}]} \quad . \quad (4)$$

Here, $[\mathbf{Q}]$ is an $N_V \times 3N_V$ matrix that projects the nodal magnetizations to the nodal point charges, whereas $[\mathbf{Z}_0]$ is an $N_V \times N_V$ matrix that describes the local correction (singularity extractions) in the potential. The matrix $[\mathbf{Q}]^T$ is the transpose of $[\mathbf{Q}]$ and it serves to map the nodal potentials to the nodal magnetostatic fields. The matrices $[\mathbf{Q}]$, $[\mathbf{Q}]^T$, and $[\mathbf{Z}_0]$ are sparse and have non-vanishing entries only for nodes that share the same tetrahedral element. The matrix $[\mathbf{Z}_l]$ is dense and it represents a mapping from N_V scalar charges to N_V scalar observers. This matrix represents the (long-range interaction) integral kernel in a canonical (point-to-point) form.

The representation in Eq. (4) is efficient and flexible in that it decouples the basis function representation from the integral kernel representation of the problem. In terms of the code developments it allows easily switching between different types of basis and testing functions or different types of the integral kernel. The matrix $[\boldsymbol{Z}_l]$ is dense and the associated matrix-vector product has a high computational cost if evaluated as a direct summation. We use the non-uniform grid interpolation method (NGIM) to evaluate this product rapidly in $O(N_v)$ operations [4-6].

The approaches discussed above are implemented on GPUs using NVIDIA CUDA programming environment [7]. A single GPU contains several hundred of stream processors, e.g. a recently released NVIDIA GeForce GTX 480 has 480 cores (double the number compared to the previous generation). Features of the GPU architecture have to be carefully taken into account when implementing the methods described above [5]. In particular, the NGIM implementations on a CPU and a GPU have significant differences.

4 Results

In this section we present results describing the performance of our method. We validated the accuracy of the code against μMAG standard problem 3 and 4. To demonstrate the code performance we present results of simulations of bit patterned media (BPM). All simulations were run on a simple desktop computer with Intel Core i7 2.66GHz CPU with 12 GB RAM and NVIDIA GeForce GTX 480 GPU (the total cost of the computer was about $2000 and the total peak power consumption was below 400 W). All computational times are quoted per time step of the LLG solver. In addition to the computational time associated with the time stepping, there is a preprocessing (or set-up) time before the simulation starts. The preprocessing time in all presented simulations varied from 0.5 sec to 20 sec.

We ran simulations of BPM for different array sizes. In all models, each island was cubic, with edge length l =12 nm. To mimic materials and patterning fluctuations, we introduced distributions in island anisotropy and position. The mean island anisotropy field and interbit spacing were 25 kOe and 12 nm, respectively. The random variations for the two quantities were 15% and 10%, respectively. In each of the following simulations all islands were initially oriented up. An external field of strength H_a = 0.91 H_K was applied downward at 1^0 to the z-axis over the entire BPM array. Here, H_a corresponded to the switching field of a single island having the mean anisotropy field. Due to the introduced distributions and magnetostatic interactions not all islands reverse under the applied field. The resulting bit-pattern for a $N \times N$ array is shown in Figure 1. The total number of bits, discretization nodes, tetrahedral elements, and the computational time per iteration are given in Table 1 for the simulated array sizes. It is evident that the computational time scales linearly with the number of elements. The absolute computational time is small

and the largest problem that can be handled is large, especially taking into account that the simulations were run on an inexpensive desktop.

It is noted that the simulated problem would be challenging for any existing solvers. For Finite Difference-based solvers accelerated by FFTs the problem would require fine grids and excessive zero padding, resulting in reduced computational time and increased memory consumption. For Finite Element/ Boundary Element-based solvers the iterative part could become slowly convergent and the surface integral part could become slow.

5 Summary

We presented a fast micromagentic solver (FastMag) for solving the Landau-Lifshiz-Gilbert equation. The solver discretizes the structure into tetrahedral elements and can handle general problems of a small or very large computational size with a high speed. The solver is implemented on Graphics Processing Units, which offer massive parallelization at a low cost, converting a simple desktop to a powerful machine matching performance of a middle range cluster. Results are presented demonstrating the efficiency of the FastMag solver.

We mention that our current code implements only the magnetostatic part on GPUs, which makes this part an order faster than the rest of the code. We are currently working on porting the remaining parts of the code to GPUs, which will reduce the quoted times by over an order. Such implementations are relatively simple and are anticipated to be completed within a month. We also intend to extend the parallelization to multi-GPU systems with the set goal of allowing the modeling of structures with a billion degrees of freedom. This will make possible the analysis and design of truly large-scale and realistic magnetic components, such as write heads, complex media, coupled oscillators.

[1] S. Li, B. Livshitz, and V. Lomakin, "Graphics processing unit accelerated O(N) micromagnetic solver," *IEEE Trans. Magn.,* Vol. 46, pp. 2373-2375, June 2010.

[2] C. W. Gardiner, *Handbook of Stochastic Methods*. Berlin: Springer, 1985.

[3] A. F. Peterson, S. L. Ray, and R. Mittra, *Computational Methods for Electromagnetics*: IEEE Press, 1997.

[4] A. Boag and B. Livshitz, "Adaptive nonuniform-grid (NG) algorithm for fast capacitance extraction," *IEEE Trans. Microwave Theory Tech.,* Vol. 54, pp. 3565-3570, September 2006.

[5] S. Li, B. Livshitz, and V. Lomakin, "Fast evaluation of Helmholtz potential on graphics processing units (GPUs)," *J. Comp. Phys.,* Vol. 229, pp. 8463-8483, 2010.

[6] B. Livshitz, A. Boag, H. N. Bertram, and V. Lomakin, "Nonuniform grid algorithm for fast calculation of magnetostatic interactions in micromagnetics," *Journal of Applied Physics,* Vol. 105, pp. 07D541 (3 pp.) April 2009.

[7] NVIDIA, "CUDA Compute Unified Device Architecture Programming Guide, V2.3," 2009.

Array Size	**# of bits**	**# of nodes**	**# of elements**	**Time**
20x20	400	25,600	64,800	0.14 sec
50x50	2,500	160,000	405,000	0.79 sec
100x100	10,000	640,000	1,620,000	3.06 sec
300x300	90,000	5,760,000	14,580,000	29.6 sec

Table 1. Results of the simulations of the BPM arrays.

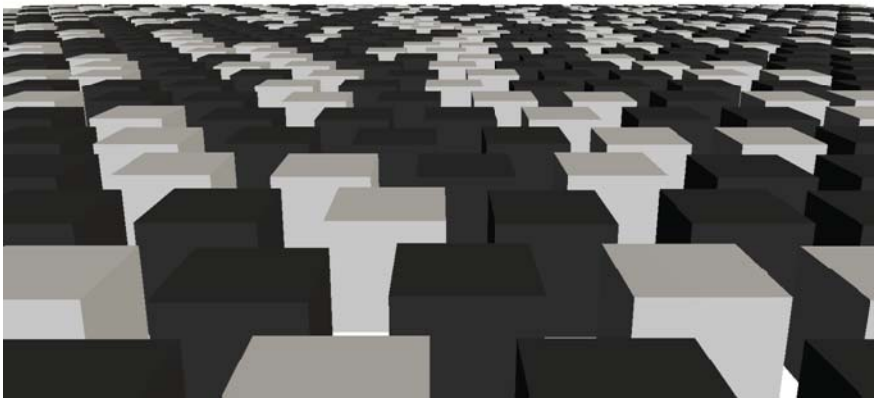

Figure 1. Resulting magnetization state pattern of a 100x100 array after reversal.