FastMag Micromagnetic Framework: Current State and Roadmap

Vitaliy Lomakin

1 Introduction

Micromagnetic solvers have a significant predictive power and are important for our ability to analyze and design magnetic devices and systems. However, micromagnetic analysis of complex and large-scale devices may be very time consuming or impossible. The computational complexity is due to the geometrical and material complexity of many structures, a large number of required discretization elements, and the numerical difficulties associated with the time integration. One example of a highly complex structure for simulations is a magnetic write head, which may have size of several tens of microns and require a mesh resolution of 5-10 nm to fully resolve all dynamic processes. More examples include a large array of Magnetic Random Access Memory elements, granular media, bit patterned media, magnonic crystals, etc. Such systems may have a large size, fine features, and highly non-uniform domains, which are required to be discretized on a fine scale. Realistic modeling of such structures can be very complex.

Here, we describe a highly flexible and efficient micromagnetic framework, FastMag. FastMag includes a set of modules that makes it well suitable for the micromagnetic analysis and design of many magnetic structures. It has a convenient interface and general meshing capabilities, can address various physics types, and runs of massively parallel Graphics Processing Unit (GPU) computer systems. The rest of the presentation summarizes our recent progress on the development of FastMag and outlines a roadmap for the FastMag development.

2 FastMag Structure and Capabilities

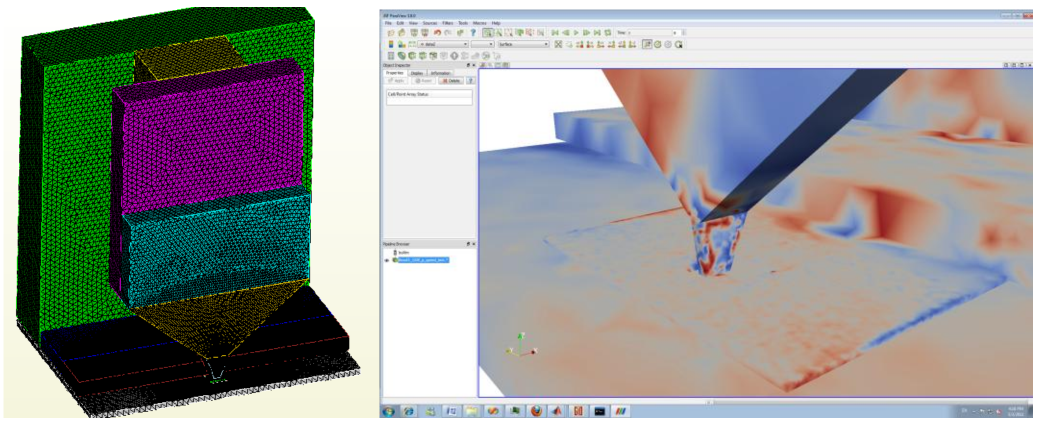

Figure 1: Cubit/Trelis model of a write head (left) and magnetization snapshot in Paraview.

2.1 Structure Definition and Discretization

FastMag is based on Finite Element Method (FEM) discretization of the computational domain. It uses Cubit (currently Trelis [-]) to create a computer model of the structure of interest. The model can be created via a Graphics User Interface (GUI) or via a Python script. A large set of geometry operations are allowed for the model creation, including Boolean operations. The structure creation is followed by the discretization into a mesh of tetrahedral elements. The tetrahedral discretization permits the analysis of highly non-uniform structures. Trelis provides several methods for the creation of meshes. It also provides tools for mesh quality assessmentsand improvements in terms of the element shape uniformity and size. Creating high quality meshes is critical for the ability to simulate complex magnetic structures. As an example, creating a mesh with ~10 million elements typically takes 5-10 minutes. Using Python allows a great flexibility in analyzing and designing structures. Python scripts can be created, which construct a structure with a set of parameters, run a micromagnetic simulation, assess the obtained results, and use this assessment to modify the structure to "close the loop" for the structure design and optimization. The simulations can be run via a GUI interface, via a command line, or via a script.

Figure 1 shows snapshots of the Cubit CAD/mesher interface and Paraview postprocessor. Figures 2-4 demonstrate some of the simulation capabilities of FastMag.

2.2 Available Modules and Capabilities

The current version of FastMag includes the following capabilities and modules:

- Solver for the Landau-Lifshitz-Gilbert equation for the study of the magnetization dynamics in general magnetic structures.
- Solver for computing energy barriers using the Nudged Elastic Band (NEB) method.
- Explicit and implicit time integration methods.
- Stochastic thermal fields in the LLG solver.
- Discretization of the structures over uniform and non-uniform tetrahedral meshes.

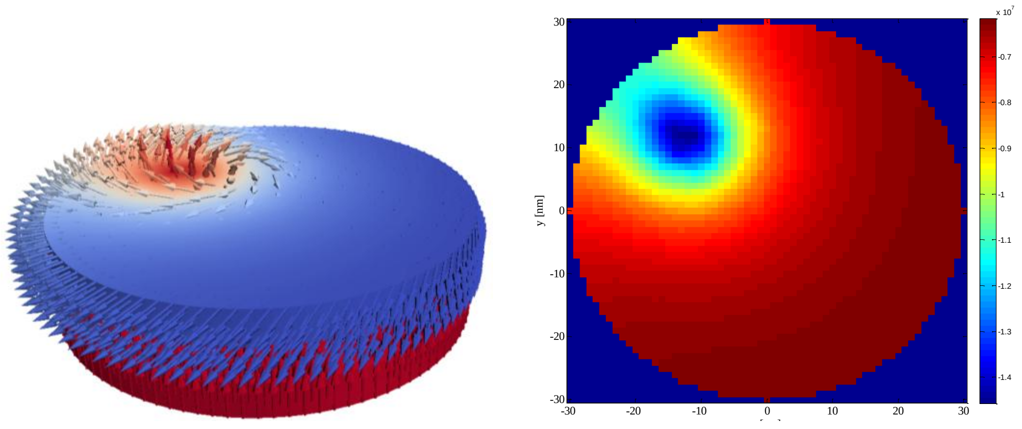

Figure 2: Vertex dynamics (lect) and corresponding currect distribution due to magnetoresistance.

- Any ferromagnetic and anti-ferromagnetic surface coupling between different volumes through common structure interfaces.
- Spin transfer torque (STT) effects in spin valves, including accounting for magnetoresistance.
- No need to discretize non-magnetic domains.
- The ability to define pinned volumes and surfaces.
- Magnetostatic fields can be computed in magnetic and non-magnetic domains.
- The speed of the solver scales linearly with the number of elements.
- The simulator runs on massively parallel GPU-based computer systems.
- The executing efficiency, fast algorithm, and GPU acceleration makes the solver able to handle complex problems on a desktop computer.

2.3 Underlying Computational Methods

In the FEM approach used in FastMag the structure under study is discretized into a mesh of tetrahedral elements, which can accurately represent generally shaped geometries. The magnetization vector is expanded over a set of interpolatory basis functions. The unknowns are the magnetization states at the nodes of the tetrahedral elements and the magnetization in the elements can be found as a summation over the basis functions. This magnetization numerical expansion is substituted into the LLG equation or equations of NEB, which leads to a system of ordinary differential equation, solved by time marching. To overcome the modeling challenges, the following points are addressed.

2.3.1 Fast Computation of the Effective Fields

One of the main costs of solving the equations of micromagnetics is the evaluation of the magnetostatic fields. In FastMag, the magnetostatic fields are computed via superposition integrals defined directly on the unstructured tetrahedral meshes. In this approach, equivalent magnetic charge densities are computed at the tetrahedron nodes, similar to the mixed potential approach in electromagnetic integral equation solvers. The scalar potential is evaluated via superposition integrals. The magnetostatic field is, then, found by numerically evaluating the gradient of the potential. The practical implementation of this approach involves several sparse matrix-vector products and a dense matrix-vector superposition product. The sparse matrices are introduced to define the differential operators, a quadrature rule that is

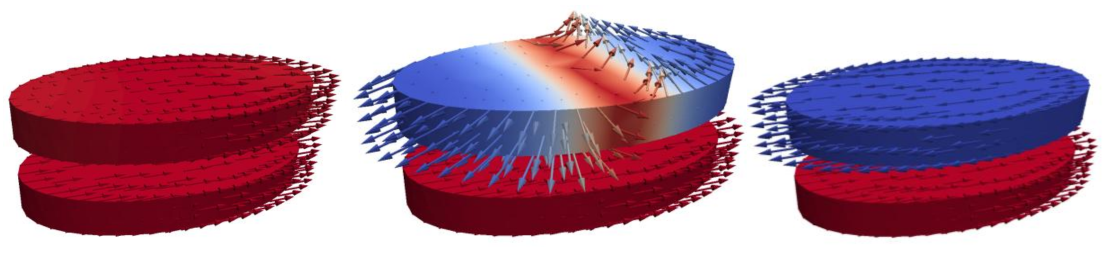
Figure 3: Switchign in-plane STT MRAM cell.

accurate for large charge-observer separations, and a singularity extraction process for analytically computing the scalar potential/field for small charge-observer separations. The dense product effectively represents a superposition summation from a number N of charges to a number N of co-located scalarobservers. The evaluation of the sparse matrix-vector products requires O(N) operations. The evaluation of the dense superposition products would require an O(N^2) operations if evaluated directly. FastMag implements a non-uniform Fast Fourier Transform (NUFFT) (also called adaptive integral method) approach, which has a computational cost of O(N logN). Both sparse and dense products are designed to be well suited to be implemented on massively parallel GPU and CPU computer systems.

The used approach for the magnetostatic field evaluation is highly efficient and flexible inin that it does not require any iterative solvers and works for non-uniform meshes. In this respect this approach provides the flexibility of conventional Finite Element/Boundary Element solvers but also has a high speed as Finite Difference solvers. The high-speed is obtained for structures of different types, including mostly volumetric structures (in which most of the mesh nodes are in volumes) and mostly surface structures (in which most of the mesh nodes are on surfaces).

Other effective fields are evaluated by defining corresponding sparse matrices and using the same sparse matrix-vector product approaches, as in the magnetostatic field evaluation case.

2.3.2 Time Integration

Many magnetic structures have strong exchange, which leads to numerical problem stiffness. The stiffness manifests itself in that the time step becomes very small and the linear solver part of implicit time integration methods becomes slowly convergent. Therefore, efficient time stepping schemes have been implemented in FastMag. In particular, the implicit schemes include the backward differentiation formula (BDF). The BDF method requires the evaluation of the numerical system Jacobian to enhance the time integration. FastMag implements a technique that allows evaluating the product of the numerical system Jacobian with the magnetization vector exactly without a need to create any matrices and it does it at the speed of the conventional effective field evaluation. The ability to execute this task allows using the BDF without a need for a linear solver preconditioner, which is important for allowing using GPUs with low memory.

2.3.3 Parallelization

Using parallel computing systems for large-scale simulations is vital for scaling computational

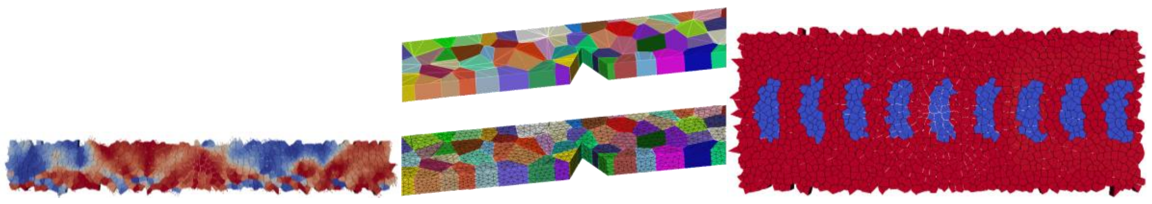
Figure 4: Modeling granular materials.

tools. Recently, new massively parallel high-performance GPU systems have emerged for applications in scientific computing, offering massive parallelization. For example, NVIDIA GeForce GTX 780 GPU at a cost of ~$500 has 2304 stream processors with the performance of 4 TFLOPs, which is much higher than any existing CPUs. Multi-GPU nodes and clusters are also available. However, not all methods or codes are easily portable to GPUs. Often directly translating a complex CPU code to GPU architectures results in a low efficiency. Features of the GPU architecture have to be carefully accounted for when developing computational methods. The complexity of the GPU architecture may make developing computational methods challenging but it opens new opportunities for creating highly efficient methods for large-scale modeling.

Most components of FastMag are built keeping parallelization in mind, including the data structure and numerical approaches. In particular, the NUFFT method has CPU-GPU speed-ups of 100-200 (single GPU vs. single CPU core). The sparse products have speed-ups of 8-20 (single GPU vs. single CPU core). Multi-GPU implementations of these methods also are available with 65-85% multi-GPU parallelization efficiency for up to 8 GPUs. Another important property of the GPU implementation of the FastMag components is low memory use, which is achieved by running many critical operations on-the-fly rather than by pre-computing and storing various coefficients as often done in CPU codes. The low memory consumption allows running large computational problems (over 100 hundred million elements) on a single GPU.

3 Roadmap of the FastMag Development

FastMag already is a general framework that allows simulating a broad range of magnetic devices and phenomena. However, we have plans of extending and improving the FastMag capabilities significantly. In the last several years a new release with significant improvements and extensions has been presented about twice a year. A similar rate of advancement is expected in the future. Specific plans are listed next.

- Additional stepping integrators.
- Streaming computing of random realizations. In this approach, multiple simulations (a stream of simulations) will be processed simultaneously on the same GPU.
- Methods to reduce stiffness in thin films and general over-discretized systems.
- Adding more physics. Among other capabilities, it includes implementing a static Maxwell solver, spin transport equations, magnetostriction and electric field effects, atomistic models, and coupling of the LLG equation with Maxwell's equations.
- Update the GUI interface + more post-processing options + more scripts for design and optimization.
- Full coupling with Python and Matlab to make the memory space of FastMag available in Matlab and Python and to allow controlling the FastMag execution through Python and Matlab.
- Coupling with SPICE for circuit models of magnetic devices, e.g. MRAM, write heads, and STT oscillators.

4 Development Team and Users

The development of FastMag has involved a significant multi-year effort. The following people have been involved in the development of the core code: Ruinan Chang, Shaojing Li, Marco Escobar, Marco Lubarda, Sidi Fu. In addition, Majd Kuteifan, Marco Menarini, and Simon Couture have joined our team more recently and have already made very important contributions. Matthew Hu has made major contributions to the development of scripts for generating FastMag input files in Cubit. Boris Livshitz was the first person in the group working on micromagnetic codes and thus made important contributions to generating ideas leading the FastMag creation.

Finally, the users of FastMag are a critical part of the development cycle. Out team appreciates and enjoys the interactions with the users. Their feedback has been, is, and will be a highly valued source of inspiration for further FastMag development.