Implementation of Iron Loss Model on Graphic Processing Units

Sajid Hussain, *Student Member, IEEE*, Rodrigo C.P. Silva, and David A. Lowther, *Senior Member, IEEE*

Department of Electrical and Computer Engineering, McGill University, Montreal, QC H3A 0E9, Canada
david.lowther@mcgill.ca

Electromagnetic design engineers are always greedy for extra computational power to finish their tasks at the earliest. It is desired to optimize numerical data processing with all the computational power available. One way is to identify a part of data with a high degree of parallelism and then process it in the graphic processing units (GPUs). GPUs are optimized to process such data efficiently and quickly on it multicore hardware. The steps involved in a finite element (FE) electromagnetic simulation are computationally very expensive. One such step is the communication between FE solver and the material loss model that takes place for every element in the mesh for each time step. This task is massively parallel and thus, can be implemented in GPUs. This work is a first step towards the implementation of material loss models in GPUs. A physics-based material model, Jiles-Atherton (JA) model, is implemented in GPU to compute the *B-H* hysteretic relationship which can be directly incorporated in FE simulation. The performance of the process is compared with the given microprocessor. It is concluded that the computational time can be greatly reduced with the help of GPUs.

Index Terms— **Graphic processing unit (GPUs), parallel processing, iron loss, Jiles-Atherton model, finite element method.**

I. Introduction

The finite Element Method (FEM) [1] is widely used to solve electrical machine design problems. If the geometrical size is large or the mesh density is high, a FE solution may be fairly expensive. It becomes even more expensive when the nonlinear hysteretic material loss model is coupled with it to compute permeability of the ferromagnetic material needed to solve Maxwell equations. As an example, if a two dimensional mesh has fifty thousand linear triangular elements; there will be the same number of material models running independently of each other. To circumvent this problem, normally, a single valued *B-H* curve is employed and the iron loss calculation is performed in the post-processing stage using curve fitting formulae.

With the rapid advancements in the multicore technology in recent years, parallel computing has played a key role in reducing the computation time. Graphic chips known as “Graphic Processing Units” (GPUs) [2] are naturally favored as data-parallel coprocessors because they have hundreds of cores which can execute computationally demanding tasks that are massively parallel in nature. They are relatively cheap, computationally powerful and energy efficient. General purpose GPU computing has been facilitated by easy-to-learn programming interfaces like CUDA [3]. GPUs have already been proven to be superior to conventional microprocessors in scientific computational problems like sparse vector matrix multiplication in iterative solvers [4].

This work is probably the first attempt to implement the iron loss models in a parallel fashion. The main objective of this work is to reduce the execution time of iron loss model. Here, we have implemented the Jiles-Atherton (JA) material loss model [5] in GPU. A large number of instances of JA model have been executed in GPU in parallel to simulate the FE simulation scenario in which each element has its own instance of loss model. In this case, the execution of the JA model in GPU provides a time gain of 225%.

II. The Jiles Atherton Model

The JA model is one of the most popular hysteresis models that can be employed into FE simulations. It is a physics-based formulation that explains the hysteresis phenomenon with the help of domain wall pinning, translation and rotation. The final model is a form of first order differential equation (shown in Eq. (1)) which can be solved iteratively.

$$\frac{dM}{dH}=\frac{1}{(1+c)}\,\frac{(M_{an}(M_S,a)-M)}{\frac{\delta k}{\mu_o}-\alpha\,(M_{an}(M_S,a)-M)}+\frac{c}{(1+c)}\,\frac{dM_{an}(M_S,a)}{dH}. \quad (1)$$

Where, $M_{an}(M_S, a)$ is the anhysteretic magnetization that is computed using Langevin’s polynomial [5], M_s, α, a, k and c are the JA model parameters. The details of these parameters, model formulation and the differential equation are given in detail in [4]. Eq. (1) is simple to implement and computationally inexpensive [6]. It only requires the five material specific model input parameters, previous value of input and output, and current value of input to predict the current output.

III. The JA Model Implementation in GPU

A. GPU Architecture

In this work, we have implemented the JA Model on NVIDIA Quadro K600 graphic card ($129 as of Nov, 2014). It has a Kepler GK107 GPU which consists of one streaming multiprocessor and has 192 SMX CUDA parallel cores running at 875 MHz. The GPU has 1024 MB of DDR3 global GPU memory, the maximum number of active threads on GPU is 2,048 and the number of 32-bit registers on GPU are 65,536. It is CUDA 3.0 compatible. The details of the CUDA versions can be seen online [7].

B. Implementation in GPU

The function that is executed on GPU is called *kernel* which in this work is the JA model itself. The input magnetic field intensity vector *H* having the length of the number of elements in the FE mesh is transferred to GPU global memory for each time step. Each entry in this vector corresponds to the magnetic field intensity $H_{element}$ of an element in the mesh. The Kernel function is executed in parallel for each element in the vector. The output magnetic flux density vector B is computed and transferred back to computer main memory (RAM). The process is repeated for N time steps (i.e. $N = 2000$).

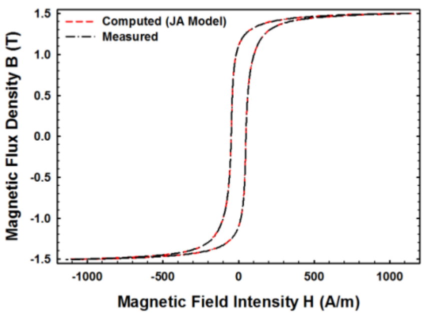
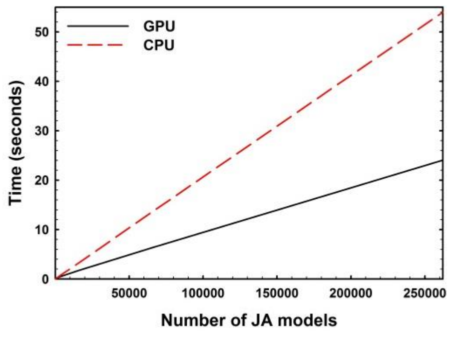
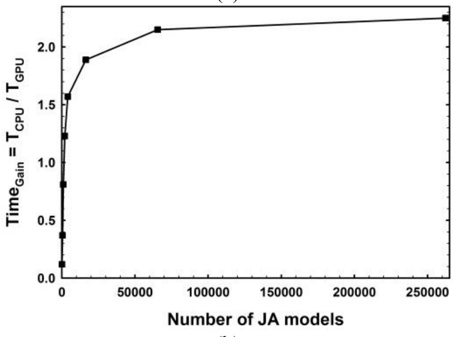


Fig. 1. Measured and computed hysteresis loop for sinusoidal H ($f = 50$ Hz). The identified JA parameters' values are $M_s = 1.229 \times 10^6$ A/m, $a = 33.7$ A/m, $\alpha = 8.77 \times 10^{-5}$, $k = 57.9$ A/m, $c = 0.05$.



(a)



(b)

Fig. 2.(a) Absolute time spent by CPU and GPU to solve x number of JA models for N = 2000 time steps (b) Time gain achieved using NVIDIA Quadro K600. All data processing is done with floating point numbers with *double* precision.

IV. Results And Discussion

In this work, we have used a Dell Precision T3610 workstation equipped with Intel Xeon® E5-1650 v2 processor running at 3.5 GHz clock and 32 GB of DDR3 RAM. The B-H loop of 35WW300 non-oriented steel was measured using the Brockhaus single sheet tester for sinusoidal flux density. Finally, the JA model parameters were identified using nonlinear least squares method [8]. The magnetic flux density B was computed for one complete time cycle for all the elements in the H vector. As an example, the B-H loop computed using JA on GPU for one of the element is compared with the measured results in Fig. 1. In Fig. 2(a), the performance of the NVIDIA Quadro K600 graphic card is compared against the sequential implementation in our Intel Xeon® processor in terms of time taken to solve a given number of JA models. The gain achieved in terms of time is shown in Fig. 2(b). It should be noted here that we have considered the total execution time in GPU. That includes, memory allocation time in GPU global memory, memory transfer time from RAM to GPU global memory, kernel execution time and memory transfer time from GPU global memory to RAM. It can be seen in Fig. 2(b) that for smaller vector lengths, memory transfer time dominates the kernel execution time, therefore, $T_{gain} < 1$. It was observed that for larger vector lengths, all the memory transfers consume less than 10 % of the total time. This is because the GPU divides the data into chunks as a multiple of active threads and performs computations thus, increasing execution time. As a proof of concept, a time gain of 225% has been achieved with this entry level GPU. In order to achieve even higher gain, the GPU burden can be shared with the CPU because CPU is free while the GPU is performing the task.

The use of more advanced GPU engines, like the Tesla K40 and K80, can further reduce the computational time of the material model in finite element (FE) simulations. These powerful coprocessors can also offer gains when using computationally expensive hysteresis models, such as the Preisach model. In this way, the hysteresis phenomenon can be efficiently incorporated into FE simulations in terms of computational time.

Acknowledgment

This work was funded by the financial grants provided by the Automotive Partnership Canada (APC) and National Science and Engineering Research Commission of Canada (NSERC).

V. References

[1] J. Jin. *The finite element method in electromagnetics*. John Wiley & Sons, 2014.

[2] J. D. Owens *et al*., "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.

[3] NVIDIA CUDA [Online]. Available: http://developer.nvidia.com/object/cuda.html

[4] M. M. Dehnavi, D. M. Fernandez, and D. Giannacopoulos, "Finite-element sparse matrix vector multiplication on graphic processing units," *Magnetics, IEEE Transactions on*, vol. 46, no. 8, pp. 2982–2985, 2010.

[5] D. C. Jiles and D. L. Atherton, "Theory of ferromagnetic hysteresis," *Journal of Magnetism and Magnetic Materials*, vol. 61, no. 12, pp. 48 – 60, 1986.

[6] S. Rosenbaum, M. Ruderman, T. Ströhla, and T. Bertram, "Use of Jiles–Atherton and Preisach hysteresis models for inverse feed-forward control," *Magnetics, IEEE Transactions on,* vol. 46, no. 12, pp. 3984-3989, 2010.

[7] CUDA [Online]. Available: http://en.wikipedia.org/wiki/CUDA.

[8] P. Kis and A. Iványi, "Parameter identification of Jiles–Atherton model with nonlinear least-square method," *Physica B: Condensed Matter*, vol. 343, no. 1–4, pp. 59–64, 2004.