5-22-2006

A Numerical Study of Dynamic Micromagnetics

He Zhang
University of New Orleans

A NUMERICAL STUDY OF DYNAMIC MICROMAGNETICS

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
The Department of Chemistry

by

He Zhang

B.S., University of Science & Technology of China,
Hefei, China, 2003

May 2006

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor, Professor Scott L. Whittenburg, for his expertise, understanding, and patience, added considerably to my graduate experience.

I would like to give my special thanks to:

- Other members of my committee, Professor John Wiley, Professor Leszek M. Malkinski and Professor Jiye Fang for their advice and encouragement on my research work.

- Dr. Zhizhong Zhao, for his selfless and fruitful discussions through this work.

- Dr. Patrick Nicholas and Dr. Feng Li for their friendships in those days.

- Many close friends for Ming Zhang, Kan Chen, Ran Liu, Lei Miao, Jianxia Zhang, Yi Wang, Jianjun Chen, Ying Long, Lifang Shi and Dr. Jibao He etc. for their caring and support through the years in University of New Orleans.

At last, I would like to dedicate this thesis to my beloved family: father, mother, sister and my wife Yan Hou.

TABLE OF CONTENTS

ABSTRACT

This research work focuses on studying the four micromagnetic fields, exchange field, anisotropy field, demagnetization field and applied field. Based on the related algorithms for programming given, a set of software for micromagnetics calculation is developed successfully in C++. Several attempts are also made to reduce the calculation complexity..

Finally this program is verified with standard problem 3 from the mumag (or micromagnetics) society. It was also used to simulate a process with a specific initial state with spins pointing each other, and the result is discussed following.

This program can also display the dynamic process of the simulation in MATLAB, which gives the information inside the material. More, this whole work is expected to be modified in order that it can easily take advantage of software platforms for parallel workstation, such as CACTUS for sharing the computer resource for further investigation.

Chapter 1 Introduction

1.1 Objectives

The aim of the micromagnetic simulation is to get information about the magnetic spin orientation of magnetic domains for micron sized objects under different conditions, such as sample size, shape or externally applied magnetic fields. Moreover, the micromagnetic simulation can also describe the extra internal magnetic configuration knowledge by comparison to experiment measurements.

1.2 History

Magnetics and micromagnetics have been explored for many centuries. The ancient Greeks and Chinese are the first to find and use naturally occurring iron ore. The key to their discovery was the observations that a magnet is able to attract other iron bearing materials. The earliest application of this permanent magnet material was in the mariner's compass before A.D.1274 [1]. During the twelfth and thirteenth centuries, this application was popular all over the world.

Around 1600 William Gilbert, a physician, published his results on observed magnetic phenomena. He was the first to apply scientific methods into the study of magnetism. Gilbert is also credited as the first to discover that the earth is a giant magnet [2]. In 1785, Charles Coulomb published the inverse square law of attraction and repulsion between magnetic poles. This is also another milestone in the field of magnetism.

In 1892, Sir James Alfred Ewing tried to explain the phenomena of ferromagnetic ordering and he coined the term hysteresis [3] [4]. Hysteresis is a property of systems that do not response the applied external forces instantly, but react slowly, or do not restore completely to their original state. In other words, the system state depends on the immediate history of the sample.

In 1907, Pierre Weiss explained the phenomena of ferromagnetic ordering qualitatively, by introducing the postulate of an internal field which tries to align the magnetic dipoles of the atoms against thermal fluctuations [3][5]. This is the first modern theory that regards magnets as composed of tiny individual magnetic domains. In the domain, the magnetization M_s is constant in magnitude and direction under some state of the system.

It's observed that even for a homogenous material, the saturation magnetization varies from one domain to another. This is the result of long-range interactions between the domains which will minimize the overall energy of the system. This understanding led to the domain wall concept introduced in 1931 by Thessen and Bitter independently [5].

Modern micromagnetics starts from 1935 with a publication by Landau and Lifshitz on the structure of a wall between two anti-parallel domains. In the 1960's, micromagnetics became a substitute for the domain wall theory. The first complete formation of the micromagnetic method was exploited by William F. Brown. A detailed book for micromagnetics was written by William Brown in 1963[4]. The concept is based on the fact that the magnitude of the saturation magnetization, M_s, is constant for each elementary cell but the orientation of the spin changes with position from cell to cell, which is in accord with the theory of Landau and Lifshitz. It was later modified by T. L. Gilbert.

In the mid-1980s the micromagnetics field developed quickly with tremendous contribution from the computer power. In modern micromagnetics, large-scale computation for realistic problems can be simulated and compared with the experimental data.

Nowadays, the numerical investigation includes two parts; one part is an energy minimization procedure which can be used to determine the nucleation field but can not necessarily predict the system state after the magnetization reversal, the other part is a dynamic approach based on the Landau-Lifshitz-Gilbert equation[3][6].

Several public software packages for micromagnetics modeling are available online. Besides the Java MicroMagnetics (http://jamm.uno.edu), one of the most popular is Object Oriented Micromagnetics Framework (OOMMF) from the National Institute of Standards

and Technology (NIST http://math.nist.gov/oommf). All the results from this simulation work are compared with oommf [7].

1.3 Properties of Micromagnetics

The magnetic moment, m, originates from the unpaired electrons in the inner electron shells of atoms. The concept of the magnetic moment was derived in the first half of the twentieth century. Mainly the moment is determined by the combined effect of the quantum mechanical exchange interaction, crystal-field interaction, relativistic spin-orbit coupling, thermal fluctuation field and Zeeman field, which will be outlined in a later section.

The total energy can be written as a sum of several energy contributions

$$E_{tot} = E_{ex} + E_{ansi} + E_{demag} + E_{zeeman} \tag{1.1}$$

The thermal energy is assumed to be Gaussian white noise which is irrelevant to the spatial coordinates or time. This work neglects calculating the thermal field during the dynamic approach to a metastable state or a local minimum energy point since it's much weaker than other four fields [8] [9].

The typical natural length scale for a micromagnetic simulation is about 10nm; meanwhile the large domain magnets may have around 1 micron diameter. Thus a typical simulation involves thousands of magnetic cells. This size of simulation can be conducted by computer power. The work reported here is to be adaptive for this range on a single workstation.

The saturation magnetization, M_s, the anisotropy constants K_1, K_2, and the exchange stiffness A are intrinsic magnetic properties (discussed below), must be

given before starting the simulation; the remanence, M, and the coercivity H_c are extrinsic properties and varied during the simulation procedure.

1.4 Applications and Future Vision

With the rapid development of high-speed computers, it's possible to make the micromagnetics simulation on the micron scale in three dimensions. This simulation method has been proposed to predict the shape effect on the induced magnetic reversal which can be used to store digital information [10] and it also provides the optimal grids for recording media, soft magnetic thin films, and allows study of the nucleation and expansion of reversed domains in hard magnetic materials [11].

In the future, a better understanding of the turbulence in the magnetic core of the earth can be simulated using a parallel calculation technique on an array of processor which would provide a simulation of the magnetic field inside and around the earth with greater detailed and accuracy [12].

Chapter 2 Micromagnetic Model

2.1 Targets

In micromagnetics, the magnetic spin is represented by a continuous function of location. The magnetic sample under study is subdivided into thousands of computational cells. The computational cell is chosen small enough so that the hundreds of spins in the cell can be reasonably modeled by a single magnetic spin for the entire cell. The magnetic spin of the cell becomes fixed in space and is only a function of the rotation of the spin. Each individual magnetization, the spin for each cell, is denoted as $\vec{M} = \vec{m} \cdot M_s$, where M_s is the saturation magnetization and $\vec{m}$ is the direction of the magnetic moment. The target of this work is to determine the magnetization directions over all the domains under various effective magnetic fields.

2.2 Numerical Micromagnetics

There are two ways to calculate the magnetization direction distribution of the elementary cells for the modeling material: the Static Method and the Dynamic Method.

2.2.1 Static Method

Static Method is also known as Brown's Static Equation [4]. In this method, each individual magnetization spin rotates gradually to the direction of the effective field at each relative position. The process of rotating the magnetization vectors subsequently in each elementary cell through the ensemble is kept going until the maximum angle is smaller than the required tolerance for all elementary cells throughout the material. The system total energy decreases for each iteration. This micromagnetic formulation was first used by LaBonte and Hubert [13].

2.2.2 Dynamic Method (Landau-Lifshitz-Gilbert Equation of Motion)

The torque formulation of the magnetization dynamics was described by Landau and Lifshitz. Later on Gilbert modified this equation by including the damping constant term as in the following equation.

$$\frac{d\vec{M}}{d\tau} = -\gamma \vec{M} \times \vec{H} - \frac{\gamma\alpha}{Ms} \vec{M} \times (\vec{M} \times \vec{H}) \tag{2.1}$$

**All the vector T related in the LLG equation are in cgs unit instead of SI unit, however, the whole LLG is reduced into the reduced unit in the later calculation part, so I just leave it here.*

$\vec{M}$ and $\vec{H}$ are vectors. M represents the spin magnetization and H is the effective field based on the combined effect of the four fields referred in *Chapter 1*. γ is the gyromagnetic ratio which is a dimensionless unit. It represents the ratio of the magnetic dipole moment to the angular momentum of an elementary particle or atomic nucleus, $\gamma = \frac{g\mu_0 e}{2m}$. The gyromagnetic ratio determines the frequency of precession of a particle in a magnetic field. α is the dimensionless damping constant in this work. It is usually set as from 0.5 to 1.0 to reach the minimized energy point. $d\tau$ is the time step.

This equation includes two terms. The first term is the precession term which rotates the spins to change the angular momentum associated with the torque due to the field $\vec{H}$ acting on the magnetic moment $\vec{M}$. As in the *figure 1*,

$$Torque: \vec{T}_1 = \vec{M} \times \vec{H} \tag{2.2}$$

The second term is an empirical damping term that directing the spin rotates to its field axis.

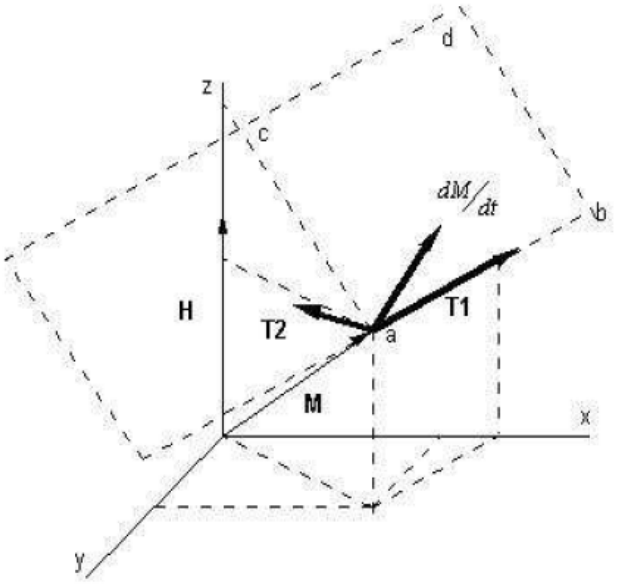


Figure 1. Related vectors in LLG equation

(Cited from "Numerical Investigation of micromagnetic structures",

Chap 3 "Theoretical Background", Attila Kakay)

All the vectors in equation (2.1) are shown in the *Figure 1*: the $\vec{M}$ and $\vec{H}$, the derivative of magnetization to time τ, $\frac{d\vec{M}}{d\tau}$, the torque T_1 in equation (2.2) and the damping term T_2.

$$\vec{T}_2 = \frac{\alpha\gamma}{M_s}\vec{M}\times(\vec{M}\times\vec{H}) \qquad (2.3)$$

The torque T_1 is perpendicular to the plane containing $\vec{M}$ and $\vec{H}$ and leads the dissipative precession of $\vec{M}$ around the effective field $\vec{H}$. The second term of vector T_2 is perpendicular to the plane of vector T_1 and $\vec{M}$, which directs the change of $\vec{M}$ towards the effective field $\vec{H}$. So the T_2 is trying to align the $\vec{M}$ along the effective field $\vec{H}$. Term

$\frac{d\vec{M}}{d\tau}$ lies in the plane (abcd) perpendicular to the $\vec{M}$, then LLG equation gives the change of $\vec{M}$ which directs itself to the effective field but by a dissipative precession.

To get a simpler LLG equation with dimensionless units, we need reduce the spin and the field by the material own saturation magnetization as following:

$$\vec{m} = \frac{\vec{M}}{M_s} \tag{2.4}$$

$$\vec{h} = \frac{\vec{H}}{M_s} \tag{2.5}$$

Obviously, the equation (2.6) comes out since the absolute magnetization, $|\vec{M}|$, is constant and is equal to the saturation magnetization.

$$|\vec{m}|^2 = 1 \tag{2.6}$$

Time is also reduced as equation (2.7)

$$t = \gamma M_s \tau \tag{2.7}$$

Further, the energy density, E, can also be reduced to

$$e = \frac{(Energy)}{\mu_0 M_s^2 V} = \frac{E}{\mu_0 M_s^2} \tag{2.8}$$

From equation (2.4) to (2.8), it should be realized that the saturation magnetization, M_s, varies for different materials, so all the reduced units must also be varied in each case.

Recalling the LLG equation in (2.1), we can rewrite it in the dimensionless form:

$$\frac{d\vec{M}}{d\tau} = -\gamma \vec{M} \times \vec{H} - \frac{\gamma\alpha}{Ms} \vec{M} \times (\vec{M} \times \vec{H})$$

$$\text{left hand side} \quad = \frac{d\vec{M}}{d\tau} = \frac{dM_s \vec{m}}{d\frac{t}{\gamma M_s}} = \gamma M_s^2 \frac{d\vec{m}}{dt}$$

$$\text{right hand side} \quad = -\gamma M_s^2 \vec{m} \times \vec{h} - \gamma\alpha M_s^2 \vec{m} \times (\vec{m} \times \vec{h})$$

$$= -\gamma M_s^2 (\vec{m} \times \vec{h} + \alpha \vec{m} \times (\vec{m} \times \vec{h}))$$

$$\Rightarrow \frac{d\vec{m}}{dt} = -\vec{m} \times \vec{h} - \alpha \vec{m} \times (\vec{m} \times \vec{h}) \qquad (2.9)$$

The discretized form for equation (2.9) is

$$\frac{\Delta\vec{m}}{\Delta t} = -\vec{m} \times \vec{h} - \alpha \vec{m} \times (\vec{m} \times \vec{h}) \qquad (2.10)$$

By the Euler's theorem, we can integrate the $\vec{m}$ by time t.

$$\vec{m}^{t+\Delta t} = \vec{m}^t + \Delta t \frac{d\vec{m}}{dt} \qquad (2.11)$$

It is worth to point out each new step result $\vec{m}^{t+\Delta t}$ need to be normalized to 1 according to the equation (2.6), because the magnitude of magnetization is constant.

$$\vec{m}^{t+\Delta t} = \frac{\vec{m}^{t+\Delta t}}{|\vec{m}^{t+\Delta t}|} \qquad (2.12)$$

For the time being, the next spin direction can be deduced from the most previous spin state based on LLG equation and Euler's theorem. By much iteration from the initial

configuration given before the calculation starting, the numerical simulation is expected to reach the metastable state where the total energy reaches a local minimum point.

To confirm the calculation reaches the metastable state, the spins must satisfy equation (2.13) according to Brown's magneto static equation:

$$\vec{M} \times \vec{H} = 0 \qquad (2.13)$$

**Equation (2.13) is also in the cgs unit.*

So in this dynamic method, a convergence criterion is also required to stop this loop based on this equation. This criterion is equal to the maximum torque over all the elementary spins under the present effective field must be less than a predefined tolerance value in order to terminate the simulation. In another words, the direct integration of the LLG equation will be carried out until the torque of each spin under the current field is smaller than the tolerance set initially. This valve value is usually between 10^{-6} and 10^{-5}. If this value set too big, the iteration may be stopped with the system at an "unstable" metastable status; if it is set to small the simulation fail to converge in a reasonable length of time.

In this work, all the micromagnetics calculations were carried out using the second method.

2.3 The Algorithms of Magnetic Energy and Fields

Recalling equation (1.1), the total energy density is

$$E_{tot} = E_{ex} + E_{ansi} + E_{demag} + E_{zeeman} \qquad (1.1)$$

This work considers only four fields to evaluate the local effective field (in SI unit), $\vec{H}$, in equation (2.1) and (2.9) by neglecting the thermal fluctuations. These four fields are exchange field (*Chapter 2.3.1*), anisotropy field (*Chapter 2.3.2*), demagnetization field or magnetostatic field (*Chapter 2.3.3*), and applied field or Zeeman field (*Chapter 2.3.4*) The thermal field can be neglected since its effects changes lead to fluctuations of the average

magnetization about 1% under normal conditions[14]. An analytical discussion about the thermal fluctuation field strength is given in the reference [9].

The differential magnetic energy can be related to the magnetic field by equation (2.14).

$$d(Energy) = \int_V B \cdot dMdV \tag{2.14}$$

For constant volume, we have the energy density E as the following,

$$\frac{d(Energy)}{dV} = \int_V B \cdot dM = E \tag{2.15}$$

Now we can derive the magnetic field from the cgs unit into the SI unit and then get the equation (2.16)

$$\frac{\partial E}{\partial \vec{M}} = -\mu_0 \vec{H} \tag{2.16}$$

For the dimensionless formula, equation (2.16) can be further deduced as the following from (2.4) and (2.5).

$$\frac{\partial E}{\partial \vec{m}} = -\mu_0 M_s^2 \vec{h}$$

Considering the equation (2.8), obtains

$$\frac{\partial e}{\partial \vec{m}} = -\vec{h} \quad \text{or} \quad \vec{h} = -\frac{1}{\mu_0 M_s^2}\frac{\partial E}{\partial \vec{m}} \tag{2.17}$$

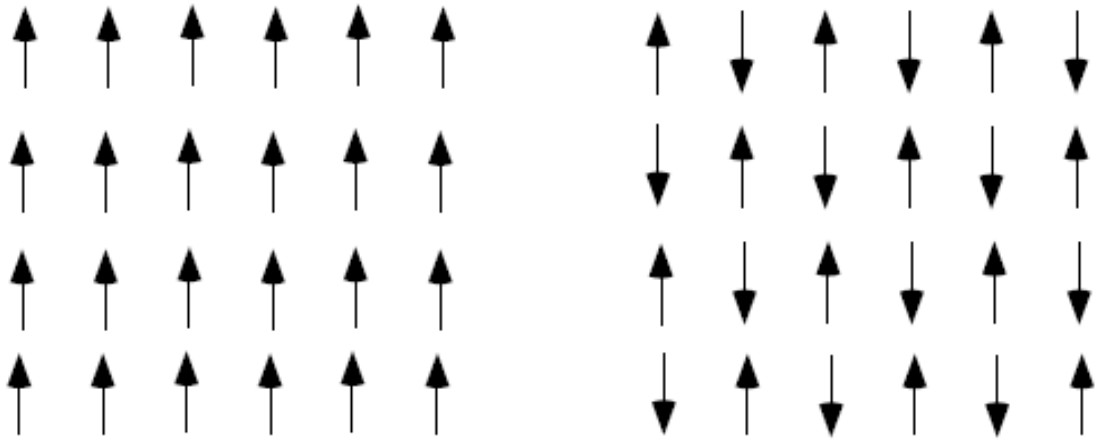
2.3.1 Exchange Field

Exchange interaction directs the cooperative magnetic ordering. The exchange energy can be scaled with the exchange integral J by Heisenberg interaction.

$$E_e = -J_{ij}\vec{S}_i\vec{S}_j$$

where $\vec{S}_i$ and $\vec{S}_j$ are the vector magnetizations of two spins on different location.

A positive J indicates ferromagnetic materials and the materials generate atomic magnetic fields aligning them parallel to external fields. This creates a greater magnetic field inside the material. A negative J gives antiferromagnetic materials, which are composed of multi-sublattices with opposite orientations.



a) *b)*

Figure 2. *a) Ferromagnetic ordering*

b) Antiferromagnetic ordering

This exchange interaction is also a behavior of the Coulomb interaction between electron charges and Pauli principle.

The exchange energy density can be calculated between two spins i and j by the equation (2.18) [15] [16],

$$E_{e,j} = A_{ij} \frac{|m_i - m_j|^2}{|R_i - R_j|^2} \tag{2.18}$$

where A is the exchange stiffness of the material under study.

Recalling the equation (2.8), (2.18) becomes a dimensionless formula as following:

$$e_{e,j} = \frac{A_{ij}}{\mu_0 M_s^2} \frac{|m_i - m_j|^2}{|R_i - R_j|^2} \tag{2.19}$$

The total energy at ith cell position is given by a continuous integration function

$$E_e = A_{ij} \int_V [(\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2] \, dV \tag{2.20}$$

where V is the volume of the sample.

In this work, the discretized integration for the ith cell by equation (2.21) is given by

$$e_e = \frac{1}{2} \sum_{j=i+1}^{N} A_{ij} \frac{| m_i - m_j |^2}{| R_i - R_j |^2} \tag{2.21}$$

where N is the total number of cells.

Since exchange energy is isotropic in nature, the calculated energy will be split into two parts evenly and distributed to the two spins (i and j) during the summation.

By the equation (2.8) and (2.17), we can derive the exchange field by the differential of equation (2.19).

$$\vec{h}_e = -\frac{\partial e_{e,j}}{\partial \vec{m}} = -\frac{A_{ij}}{\mu_0 M_s^2 | R_i - R_j |^2} \frac{\partial | \vec{m}_j - \vec{m}_i |^2}{\partial \vec{m}}$$

$$\vec{h}_e = -\frac{2A_{ij}}{\mu_0 M_s^2 | R_i - R_j |^2} (\vec{m}_i - \vec{m}_j) \tag{2.22}$$

The reduced exchange constant a_{ij} is introduced for each domain, which is calculated in the initial part of the simulation and stored separately as a public variable, as equation (2.23). All the a_{ij}'s are equal in the homogenous case.

$$a_{ij} = -\frac{2A_{ij}}{\mu_0 M_s^2 | R_i - R_j |^2} \tag{2.23}$$

Putting the exchange constant into equation (2.21), the reduced exchange energy is calculated by

$$e_e = \frac{1}{4}\sum_{j=i+1}^{N} a_{ij} \mid m_i - m_j \mid^2 \qquad (2.24)$$

2.3.2 Anisotropy Field

Anisotropy energy depends on the spin orientation of the magnetization with respect to the crystallographic axes of the material. This field results from the spin-orbit interaction [17]. There is a strong coupling between the spin and the orbital angular momentum at atomic level. Because the atomic orbital is usually not spherical but more elliptical, the spin-orbit coupling causes the spin to prefer to lie along some crystallographic direction. This direction is termed the *easy axis* for the magnetization. The rotation of the magnetization away from these easy axes is taken account for by the anisotropy energy and field. Also, according the number of easy axis in the lattice, lattice structures can be classified as uniaxial, biaxial, cubic and higher order anisotropy.

2.3.2.1 Uniaxial Anisotropy

All the spins prefer to align in on direction parallel to the only easy axis in this case to minimize the energy. The general form for the energy density can be written as equation (2.25) as follows:

$$E_{a\text{Uniaxial}} = \sum_j K_j \sin^{2j}\theta$$

$$\cong K_0 + K_1 \sin^2\theta + K_2 \sin^4\theta \qquad (2.25)$$

where K_j's are the anisotropy constants which are temperature dependent; and θ is the polar angle for the magnetization, $\vec{M}$, relative to the easy axis.

The index for the "sinθ" term is always an even number because the energy surface is symmetric upon rotation. This work neglects the higher order for j larger than 2. Also, K_0 is the zeroth order energy for all the spins are oriented along the crystalline axis.

Recalling the equation (2.4),

$$\vec{m} = \frac{\vec{M}}{M_s} \tag{2.4}$$

Since the $\vec{m}$ just represents the spin direction, introducing the unitless easy axis $\vec{a}$, the "$\sin^2\theta$" term can be expressed in two different ways using trigonometric identities

$$\vec{m} \cdot \vec{a} = |m||a| \cos\theta = \cos\theta$$

$$\sin^2\theta = 1 - (\vec{m} \cdot \vec{a})^2 \tag{2.26}$$

Bringing this equation back into (2.25), (2.27) is obtained.

$$E_{a\text{Uniaxial}} = K_0 + K_1[1 - (\vec{m} \cdot \vec{a})^2] + K_2[1 - (\vec{m} \cdot \vec{a})^2]^2 \tag{2.27}$$

As discussed before, the K_0 cannot be negative, so when is K_1 positive, the energy minimum occurs when the spin is parallel to the easy axis, so K_0 will be zero. Equation (2.27) becomes,

$$E_{a\text{Uniaxial}} = K_1[1 - (\vec{m} \cdot \vec{a})^2] + K_2[1 - (\vec{m} \cdot \vec{a})^2]^2 \qquad (K_1 \text{ non-negative}) \tag{2.28}$$

When the K_1 negative, the energy minimum will occur when the direction of the spin is perpendicular to the easy axis, where $K_0 = -K_1$, so equation (2.27) becomes,

$$E_{a\text{Uniaxial}} = -K_1 + K_1[1 - (\vec{m} \cdot \vec{a})^2] + K_2[1 - (\vec{m} \cdot \vec{a})^2]^2$$

$$E_{a\text{Uniaxial}} = -K_1(\vec{m} \cdot \vec{a})^2 + K_2[1 - (\vec{m} \cdot \vec{a})^2]^2 \quad (K_1 \text{ negative}) \tag{2.29}$$

By equation (2.17)

$$\vec{h} = -\frac{1}{\mu_0 M_s^2}\frac{\partial E}{\partial \vec{m}} \tag{2.17}$$

The dimensionless anisotropy field is expressed by differential (2.28) and (2.29) in terms of $\vec{m}$ and $\vec{a}$.

From equation (2.28):

$$\vec{h}_{a\text{Uniaxial}} = -\frac{1}{\mu_0 M_s^2}\frac{\partial\{K_1[1-(\vec{m}\cdot\vec{a})^2]+K_2[1-(\vec{m}\cdot\vec{a})^2]^2\}}{\partial\vec{m}}$$

$$= -\frac{1}{\mu_0 M_s^2}\{2K_1[-(\vec{m}\cdot\vec{a})\vec{a}]+2K_2[1-(\vec{m}\cdot\vec{a})^2][-2(\vec{m}\cdot\vec{a})\vec{a}]\}$$

$$= \frac{(\vec{m}\cdot\vec{a})\vec{a}}{\mu_0 M_s^2}\{2K_1+4K_2[1-(\vec{m}\cdot\vec{a})^2]\}$$

$$= \{\frac{2K_1}{\mu_0 M_s^2}+\frac{4K_2}{\mu_0 M_s^2}[1-(\vec{m}\cdot\vec{a})^2]\}(\vec{m}\cdot\vec{a})\vec{a}$$

From equation (2.29):

$$\vec{h}_{a\text{Uniaxial}} = -\frac{1}{\mu_0 M_s^2}\frac{\partial\{-K_1(\vec{m}\cdot\vec{a})^2+K_2[1-(\vec{m}\cdot\vec{a})^2]^2\}}{\partial\vec{m}}$$

$$= -\frac{1}{\mu_0 M_s^2}\{-2K_1[(\vec{m}\cdot\vec{a})\vec{a}]+2K_2[1-(\vec{m}\cdot\vec{a})^2][-2(\vec{m}\cdot\vec{a})\vec{a}]\}$$

$$= \frac{(\vec{m}\cdot\vec{a})\vec{a}}{\mu_0 M_s^2}\{2K_1+4K_2[1-(\vec{m}\cdot\vec{a})^2]\}$$

$$= \{\frac{2K_1}{\mu_0 M_s^2}+\frac{4K_2}{\mu_0 M_s^2}[1-(\vec{m}\cdot\vec{a})^2]\}(\vec{m}\cdot\vec{a})\vec{a}$$

Therefore, regardless of the sign of the first anisotropy constant, the formula for the anisotropy field is the same. We simplify the anisotropy field as the equation (2.30).

$$h_{a\text{Uniaxial}} = \{k_1 + 2k_2[1-(\vec{m}\cdot\vec{a})^2]\}(\vec{m}\cdot\vec{a})\vec{a} \tag{2.30}$$

where $k_i = \dfrac{2K_i}{\mu_0 M_s^2}$ i = 1, 2 (2.31)

According to the above equation, the energy density can also be simplified from (2.28) and (2.29) to obtain the dimensionless e_a by equation (2.8).

$$e_{aUniaxia} = \frac{E_{aUniaxial}}{\mu_0 M_s^2} = \begin{cases} \dfrac{K_1[1-(\vec{m}\cdot\vec{a})^2] + K_2[1-(\vec{m}\cdot\vec{a})^2]^2}{\mu_0 M_s^2} & K_1 \geq 0 \\ \dfrac{-K_1(\vec{m}\cdot\vec{a})^2 + K_2[1-(\vec{m}\cdot\vec{a})^2]^2}{\mu_0 M_s^2} & K_1 < 0 \end{cases}$$

$$\Rightarrow e_{aUniaxia} = \begin{cases} \dfrac{k_1[1-(\vec{m}\cdot\vec{a})^2] + k_2[1-(\vec{m}\cdot\vec{a})^2]^2}{2} & K_1 \geq 0 \\ \dfrac{-k_1(\vec{m}\cdot\vec{a})^2 + k_2[1-(\vec{m}\cdot\vec{a})^2]^2}{2} & K_1 < 0 \end{cases} \tag{2.32}$$

2.3.2.2 Cubic Anisotropy

The cubic anisotropy field has three orthogonal crystal easy axes, $\vec{a}$ (parallel to x axis), $\vec{b}$ (parallel to y axis), $\vec{c}$ (parallel to z axis), respectfully.

$$\vec{a} = \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix} \qquad \vec{b} = \begin{vmatrix} 0 \\ 1 \\ 0 \end{vmatrix} \qquad \vec{c} = \begin{vmatrix} 0 \\ 0 \\ 1 \end{vmatrix}$$

The cubic anisotropy energy density is basically expressed by equation (2.33).

$$E_{aCubic} = K_1(m_x^2 m_y^2 + m_x^2 m_z^2 + m_y^2 m_z^2) + K_2 m_x^2 m_y^2 m_z^2 \tag{2.33}$$

Where m_x, m_y, m_z are the magnetization components along the $\vec{a}$, $\vec{b}$, and $\vec{c}$ axes, respectively. K_i is the i-*th* order anisotropy constant. Here the higher order anisotropy constant terms will also be neglected.

Recalling equation (2.26), (2.33) can be rewritten as the following.

$$E_{aCubic} = K_1(\cos^2\alpha\cos^2\beta + \cos^2\alpha\cos^2\gamma + \cos^2\beta\cos^2\gamma) + K_2\cos^2\alpha\cos^2\beta\cos^2\gamma$$

$$E_{aCubic} = K_1[(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{b})^2 + (\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{c})^2 + (\vec{m}\cdot\vec{b})^2(\vec{m}\cdot\vec{c})^2] + K_2(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{b})^2(\vec{m}\cdot\vec{c})^2$$

(2.34)

Refer to equation (2.8) to get the dimensionless formula for cubic energy density.

$$e_{aCubic} = \frac{K_1[(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{b})^2 + (\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{c})^2 + (\vec{m}\cdot\vec{b})^2(\vec{m}\cdot\vec{c})^2] + K_2(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{b})^2(\vec{m}\cdot\vec{c})^2}{\mu_0 M_s^2}$$

$$= \frac{k_1}{2}[(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{b})^2 + (\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{c})^2 + (\vec{m}\cdot\vec{b})^2(\vec{m}\cdot\vec{c})^2] + \frac{k_2}{2}(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{b})^2(\vec{m}\cdot\vec{c})^2$$

(2.35)

where k_i also obeys the rule of equation (2.31).

Same way as uniaxial the cubic anisotropy field is obtained by equation (2.17).

$$\vec{h}_{aCubic} = -\frac{\partial\{\frac{k_1}{2}[(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{b})^2 + (\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{c})^2 + (\vec{m}\cdot\vec{b})^2(\vec{m}\cdot\vec{c})^2] + \frac{k_2}{2}(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{b})^2(\vec{m}\cdot\vec{c})^2\}}{\partial\vec{m}}$$

$$= -\frac{k_1}{2}[2\vec{a}(\vec{m}\cdot\vec{b}) + 2\vec{b}(\vec{m}\cdot\vec{a}) + 2\vec{a}(\vec{m}\cdot\vec{c}) + 2\vec{c}(\vec{m}\cdot\vec{a}) + 2\vec{b}(\vec{m}\cdot\vec{c}) + 2\vec{c}(\vec{m}\cdot\vec{b})]$$

$$-\frac{k_2}{2}[2\vec{a}(\vec{m}\cdot\vec{b})^2(\vec{m}\cdot\vec{c})^2 + 2\vec{b}(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{c})^2 + 2\vec{c}(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{b})^2]$$

$$= -\{k_1[\vec{a}(\vec{m}\cdot\vec{b}) + \vec{b}(\vec{m}\cdot\vec{a}) + \vec{a}(\vec{m}\cdot\vec{c}) + \vec{c}(\vec{m}\cdot\vec{a}) + \vec{b}(\vec{m}\cdot\vec{c}) + \vec{c}(\vec{m}\cdot\vec{b})]$$

$$+ k_2[\vec{a}(\vec{m}\cdot\vec{b})^2(\vec{m}\cdot\vec{c})^2 + \vec{b}(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{c})^2 + \vec{c}(\vec{m}\cdot\vec{a})^2(\vec{m}\cdot\vec{b})^2]\}$$

Now considering the dot product of the magnetization and the axis,

$$\vec{m} = \begin{vmatrix} m_x \\ m_y \\ m_z \end{vmatrix} \quad \text{and} \quad \vec{a} = \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix} \quad \vec{b} = \begin{vmatrix} 0 \\ 1 \\ 0 \end{vmatrix} \quad \vec{c} = \begin{vmatrix} 0 \\ 0 \\ 1 \end{vmatrix}$$

$$\vec{m}\cdot\vec{a} = m_x \times 1 + m_y \times 0 + m_z \times 0 = m_x$$

$$\vec{m}\cdot\vec{b} = m_x \times 0 + m_y \times 1 + m_z \times 0 = m_y$$

$$\vec{m}\cdot\vec{c} = m_x \times 0 + m_y \times 0 + m_z \times 1 = m_z$$

Calculating cubic anisotropy from energy density gives,

$$\begin{aligned}\vec{h}_{aCubic} &= -\{k_1[\vec{a}m_y^{\,2} + \vec{b}m_x^{\,2} + \vec{a}m_z^{\,2} + \vec{c}m_x^{\,2} + \vec{b}m_z^{\,2} + \vec{c}m_y^{\,2}] \\ &+ k_2[\vec{a}m_y^{\,2}m_z^{\,2} + \vec{b}m_x^{\,2}m_z^{\,2} + \vec{c}m_x^{\,2}m_y^{\,2}]\}\end{aligned}$$

$$\begin{aligned}\vec{h}_{aCubic} &= -\vec{a}[k_1(m_y^{\,2} + m_z^{\,2}) + k_2 m_y^{\,2}m_z^{\,2}] \\ &\quad -\vec{b}[k_1(m_x^{\,2} + m_z^{\,2}) + k_2 m_x^{\,2}m_z^{\,2}] \\ &\quad -\vec{c}[k_1(m_x^{\,2} + m_y^{\,2}) + k_2 m_x^{\,2}m_y^{\,2}]\end{aligned} \tag{2.36}$$

Here we have presented expressions for uniaxial and cubic anisotropy. Other anisotropy types are less common and not carried out in this work.

2.3.3 Magnetostatic or Demagnetization Field

The demagnetization energy will be minimized when the spins stay in a “head-tail” configuration, which also termed as vortex state. *Figure 3* is a typical result when only the demagnetization field is included in the calculation.

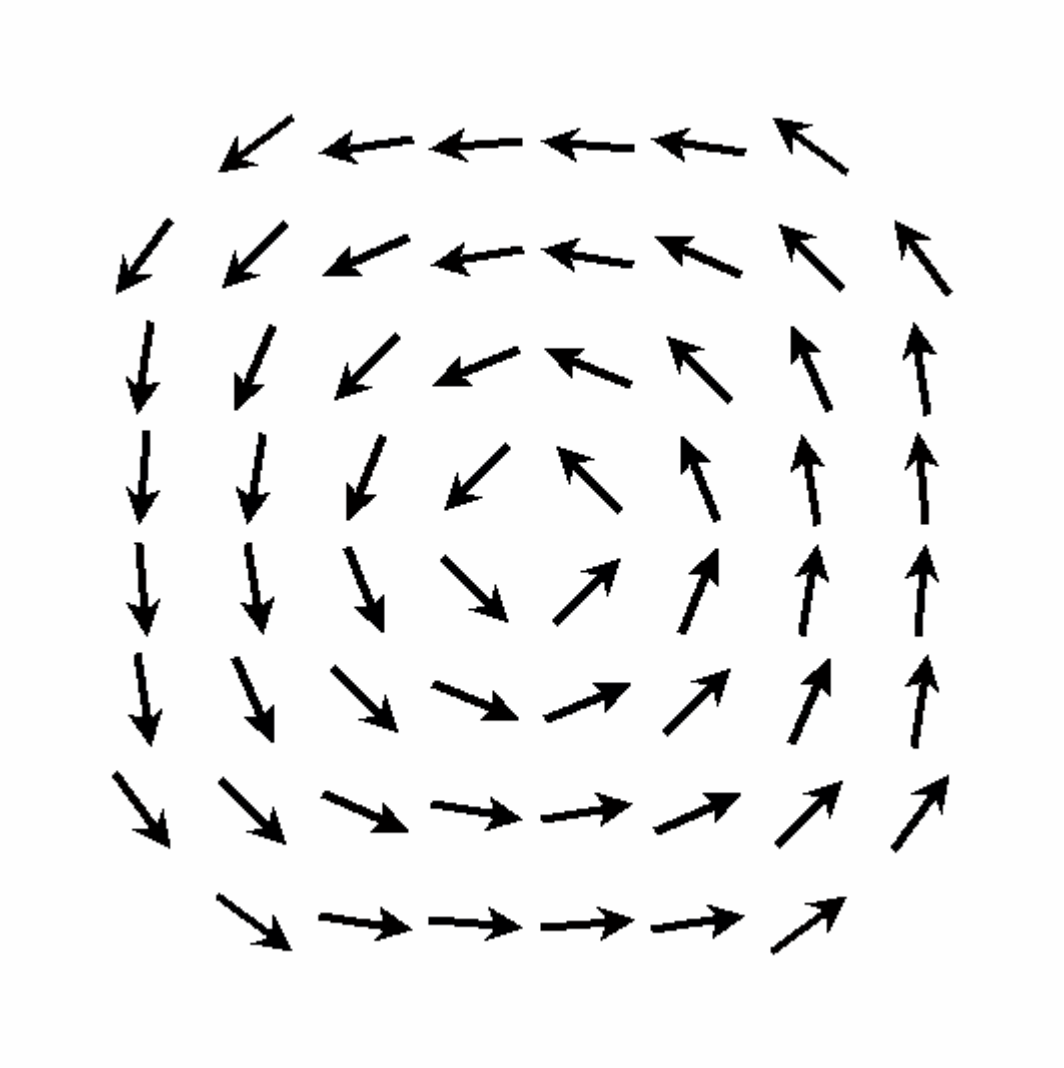


Figure 3. "head-tail" configuration resulting by including demagnetization field only

2.3.3.1 Theory for Demagnetization Field and Energy calculation

The magnetostatic self energy is given in equation (2.37) [6] [18].

$$E_{d,ij} = \frac{1}{2}\mu_0 \widetilde{N}_{ij} \vec{M}_j \vec{M}_i$$

By equation (2.16), the magnetostatic field given as

$$\vec{H}_{d,ij} = -\widetilde{N}_{ij} \cdot \vec{M}_i$$

$\widetilde{N}_{ij}$ is demagnetizing tensor depending on the shape of the sample and depends on the relative position for two spins i and j.

Reducing the demagnetization energy and field into dimensionless by equation (2.4), (2.5) and (2.8), gives

$$e_{d,ij} = \frac{1}{2}\mu_0 \widetilde{N}_{ij} \vec{m}_j \vec{m}_i \tag{2.37}$$

$$\vec{h}_{d,ij} = -\vec{N}_{ij} \cdot \vec{m}_j \tag{2.38}$$

The assumption for equation (2.37) and (2.38) is that the material has uniformly distributed saturation magnetization, M_s. Otherwise multiple $M_{s,j}/M_{s,i}$ must be added on the right hand side.

There several methods to evaluate this tensor such as by the Fast Multipole method [19] [20] [21], but the most accurate was given by Andrew Newell [22] in 1993. In Cartesian coordinates, the tensor for dipole-dipole approximation is given by the following,

$$\vec{N} = \begin{vmatrix} N_{xx} & N_{xy} & N_{xz} \\ N_{yx} & N_{yy} & N_{yz} \\ N_{zx} & N_{zy} & N_{zz} \end{vmatrix} \tag{2.39}$$

The calculation for each element in this matrix is divided by two parts. This first step is to calculate the element on the diagonal N_{xx}, N_{yy}, N_{zz}:

$$N_{xx}(X,Y,Z) = \frac{1}{4\pi V}[2F(X,Y,Z) - F(X+\Delta x,Y,Z) - F(X-\Delta x,Y,Z)] \tag{2.40}$$

where X, Y, Z are the coordinates differences between two spins; Δx, Δy, Δz are the cubic cell side length in this work; V is the cell volume as $\Delta x \cdot \Delta y \cdot \Delta z = (\Delta x)^3$ for cube.

F(X, Y, Z) is given by equation (2.40) recursively.

$$F(X,Y,Z) = F_1(X,Y+\Delta y,Z+\Delta z) - F_1(X,Y,Z+\Delta z) - F_1(X,Y+\Delta y,Z) + F_1(X,Y,Z)$$

(2.41)

F_1(X, Y, Z) is given by equation (2.41) recursively.

$$F_1(X,Y,Z) = F_2(X,Y,Z) - F_2(X,Y-\Delta y,Z) - F_2(X,Y,Z-\Delta z) + F_2(X,Y-\Delta y,Z-\Delta z)$$

(2.42)

F_1(X, Y, Z) is given by equation (2.42) recursively.

$$F_2(X,Y,Z) = f(X,Y,Z) - f(X,0,Z) - f(X,Y,0) + f(X,0,0)$$

(2.43)

f(X, Y, Z) is given by equation (2.43) recursively.

$$f(X,Y,Z) = \frac{Y}{2}(Z^2 - X^2)\sin^{-1}\left(\frac{Y}{\sqrt{X^2+Z^2}}\right) + \frac{Z}{2}(Y^2 - X^2)\sin^{-1}\left(\frac{Z}{\sqrt{Y^2+Z^2}}\right)$$
$$- XYZ\tan^{-1}\left(\frac{yz}{xR}\right) + \frac{1}{6}(2x^2 - y^2 - z^2)R$$

(2.44)

where $R = \sqrt{X^2+Y^2+Z^2}$.

So far, we have the first element N_{xx} in the object matrix. This work calculates the other two diagonal elements by technically permuting variables by calling the same function N_{xx}(X, Y, Z, dx, dy, dz). For instance, to calculate N_{xx} by calling $N_{xx}(x_i, y_i, z_i, dx, dy, dz)$, then get the N_{yy} by calling $N_{xx}(y_i, x_i, z_i,, dy, dx, dz)$ and N_{zz} by calling $N_{xx}(z_i, y_i, x_i, dz, dy, dx)$.

The second step is to calculate the elements in the tensor matrix off-diagonal. Component N_{xy} is given as:

$$N_{xy}(X,Y,Z)=\frac{1}{4\pi V}[G(X,Y,Z)-G(X-\Delta x,Y,Z)-G(X,Y+\Delta y,Z)+G(X-\Delta x,Y+\Delta y,Z)] \tag{2.45}$$

G(X, Y, Z) is given by equation (2.46) recursively.

$$G(X,Y,Z)=G_1(X,Y,Z)-G_1(X,Y-\Delta y,Z)-G_1(X,Y,Z-\Delta z)+G_1(X,Y-\Delta y,Z-\Delta z) \tag{2.46}$$

G(X, Y, Z) is given by equation (2.47) recursively.

$$G_1(X,Y,Z)=G_2(X+\Delta x,Y,Z+\Delta z)-G_2(X+\Delta x,Y,Z)-G_2(X,Y,Z+\Delta z)+G_2(X,Y,Z) \tag{2.47}$$

G(X, Y, Z) is given by equation (2.48) recursively.

$$G_2(X,Y,Z)=g(X,Y,Z)-g(X,Y,0) \tag{2.48}$$

(X, Y, Z) is given by equation (2.49) recursively.

$$\begin{aligned} g(X,Y,Z)=&(XYZ)\sin^{-1}\left(\frac{Z}{\sqrt{X^2+Y^2}}\right)+\frac{Y}{6}(3Z^2-Y^2)\sin^{-1}\left(\frac{X}{\sqrt{Y^2+Z^2}}\right)\\ &+\frac{X}{6}(3Z^2-X^2)\sin^{-1}\left(\frac{Y}{\sqrt{X^2+Z^2}}\right)-\frac{XY^2}{2}\tan^{-1}\left(\frac{XZ}{YR}\right)-\frac{ZX^2}{2}\tan^{-1}\left(\frac{YZ}{XR}\right)\\ &-\frac{X^3}{6}\tan^{-1}\left(\frac{XY}{ZR}\right)-\frac{XYR}{3} \end{aligned} \tag{2.49}$$

Using the same permuting technique as for diagonal elements, we can calculate all the other five off-diagonal components.

2.3.3.2 Techniques in Demagnetization calculation

The most outstanding feature for this field is that it has a natural long range interaction for all the cells in the magnetic sample. This suggests all the spins will be correlated with each other through the demagnetization interaction. For example, in a computational box containing *n* unit cells, the magnetostatic energy term for each cell will include interaction with all the other cells, so the total calculation will involve $O(n^2)$ complexity. In contrast, for exchange and anisotropy energy, the calculation only involves a single spin and is $O(n)$. Obviously the demagnetizing energy term makes the heaviest demands on the CPU and memory.

From equation (2.44) and (2.49), the demag tensors have some symmetry characteristics. All the diagonal elements are even functions, and the off-diagonal elements are odd function. For example, N_{xx} is an even function of (X, X):

$$N_{xx}(X,Y,Z) = N_{xx}(-X,Y,Z) = N_{xx}(X,-Y,Z) = N_{xx}(X,Y,-Z)$$
$$= N_{xx}(-X,-Y,Z) = N_{xx}(-X,Y,-Z) = N_{xx}(X,-Y,-Z) = N_{xx}(-X,-Y,-Z)$$

(2.50)

For N_{xy} which is odd function of (X, Y):

$$N_{xy}(X,Y,Z) = -N_{xy}(-X,Y,Z) = -N_{xy}(X,-Y,Z) = N_{xy}(X,Y,-Z)$$
$$= N_{xy}(-X,-Y,Z) = -N_{xy}(-X,Y,-Z) = -N_{xy}(X,-Y,-Z) = N_{xy}(-X,-Y,-Z)$$

(2.51)

According to this feature, two methods were tried to reduce the calculation complexity.

Technique 1: This work initially calculates all the tensors only for one corner cell and stores this array as public variables. As shown in *figure 4*, this program calculates the cell[0]'s demag tensors from cell[i] (i varies from "0" to "24"), and stores the result as "public tensor[x][y]", a two dimension array, where "x" and "y" are the sequence in x and y axes.

20	21	22	23	24
15	16	17	18	19
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

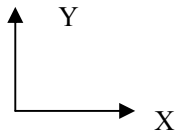


Figure 4. *An example for technique 1 applied in demagnetization field calculation*

Further, when the program requested the new tensor for any two cell[i] and cell[j] (i, j are in the range of "0" to "24"), another function will select the proper tensor from the previous public array based on the tensor's properties: all elements are odd functions of their subjects.

If (X<0) then $N_{XY} = -N_{XY}$; $N_{XZ} = -N_{XZ}$;

If (Y<0) then $N_{XY} = -N_{XY}$; $N_{YZ} = -N_{YZ}$;

If (Z<0) then $N_{XZ} = -N_{XZ}$; $N_{YZ} = -N_{YZ}$;

(2.52)

For instance, to pick up the proper tensor from "public tensor[x][y]" for cell[16] and cell[8]. Because the difference in x-coordinate is -2, and difference in y-coordinate is 2 for these two cells, the tensors must have the same absolute values as the sensor for cell[0] and cell[12], "public tensor[2][2]". More, by the algorithm of equation (2.52), only the N_{xy} and N_{xz} change sign while other four components keep the same. Since the calculated public array is from a corner cell, its range will cover the combinations for any two cells.

Using this technique, the calculation of the demagnetization tensors is done only once at the beginning of the calculation and is not done during each iteration of the LLG loop.

This technique doesn't reduce the complexity, which is still *O*(n), however it reduces the calculations in each LLG iteration.

Technique 2: The Fast Fourier Transform (FFT) is a way to compute the Fourier transform of a convolution in time O(nlogn) instead of $O(n^2)$ using the direct method. The FFT can be used with when the number of cells, n, is a power of 2.

The premise for the FFT method is the Convolution Theorem. f(r) and h(r) are two discrete sequences and they have the convolution relationship such as given equation (2.53).

$$f * h = \sum_{r'}^{r} g(r') f(r - r') \tag{2.53}$$

It has been proven that if two functions are convolved in the spatial domain, their FFT transformed functions in the momentum, or q domain, are a simple product. This product is only relative to O(n) complexity, however the FFT transform is $O(\log_2 n)$ complexity, so the total complexity is reduced from $O(n^2)$ to $O(n \log_2 n)$. Note that the demagnetization calculation involves a convolution of the demagnetization tensor, N_{ij}, and the magnetization vector, M_j, as shown in Equation (2.38). Thus, calculation of the demagnetization field, $H_{d,ij}$, can be speeded up by using the FFT method.

The main idea of this technique is given as the following frame, *figure 5.*

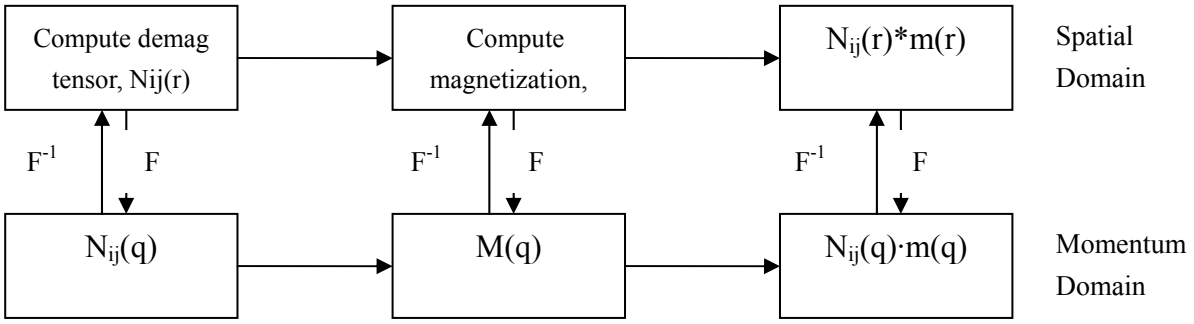


Figure 5. The FFT method in micromagnetics

Recall equation (2.38) which calculate the demagnetization field from cell j on cell i,

$$\vec{h}_{d,ij} = -\vec{N}_{ij} \cdot \vec{m}_j$$

To get the total demagnetization field of the i-*th* cell from the whole box, we need sum up equation (2.38) over all cells, so it comes out as,

$$\vec{h}_{d,i} = -\sum_{j}^{N} \vec{N}_{ij} \cdot \vec{m}_j \tag{2.54}$$

Expending the vectors, we get three individual equations for three dimensions.

$$h_{d,i}.x = -\sum_{j}^{N} (Nxx_{ij} \times m_j.x + Nxy_{ij} \times m_j.y + Nxz_{ij} \times m_j.z) \tag{2.55}$$

$$h_{d,i}.y = -\sum_{j}^{N} (Nxy_{ij} \times m_j.x + Nyy_{ij} \times m_j.y + Nyz_{ij} \times m_j.z) \tag{2.56}$$

$$h_{d,i}.z = -\sum_{j}^{N} (Nxz_{ij} \times m_j.x + Nyz_{ij} \times m_j.y + Nzz_{ij} \times m_j.z) \tag{2.57}$$

The application of the FFT method to micromagnetics, therefore, is accomplished as follows. During the initialization of the program the demag tensor elements are calculated in the spatial domain and FFT transformed to the momentum domain and stored. During each iteration in the LLG method the current estimate for the magnetization components, m_{jx}, m_{jy} and m_{jz}, are FFT transformed into the momentum domain and then the product of N_{ij} and m_j is carried out in the momentum domain. The demag field is therefore obtained in the momentum domain as given in equation (2.55), (2.56) and (2.57). The demag field is then inverse FFT transformed back into the spatial domain so that it is in the same domain as the other fields and energy terms.

In spite of the fact that a FFT code is generated and verified successfully, the FFT method was not carried out in this work. The FFT code in C++ is attached in *Appendix 3*.

2.3.4 Zeeman Field

Zeeman field arises from any applied magnetic field on the system. The Zeeman field acts on each spin. The magnitude and direction of the applied field is given by users during

the initialization part of the program. The dimensionless field is given by the equation (2.58).

$$\vec{h}_z = \frac{\vec{H}_z}{M_s} \tag{2.58}$$

Applying integral on equation (2.17)

$$\frac{\partial e}{\partial \vec{m}} = -\vec{h} \tag{2.17}$$

$$\int \partial e = -\vec{h} \cdot \int \partial \vec{m}$$

$$\Rightarrow e = -\vec{h} \cdot \vec{m} \tag{2.59}$$

Chapter 3 Micromagnetics Program Scheme and Verification

3.1 Micromagnetics Program Scheme

Most this code is done by Microsoft Visual Studio, VC++ except the display part by MATLAB 7.0. The following figure is the flow chart of this work.

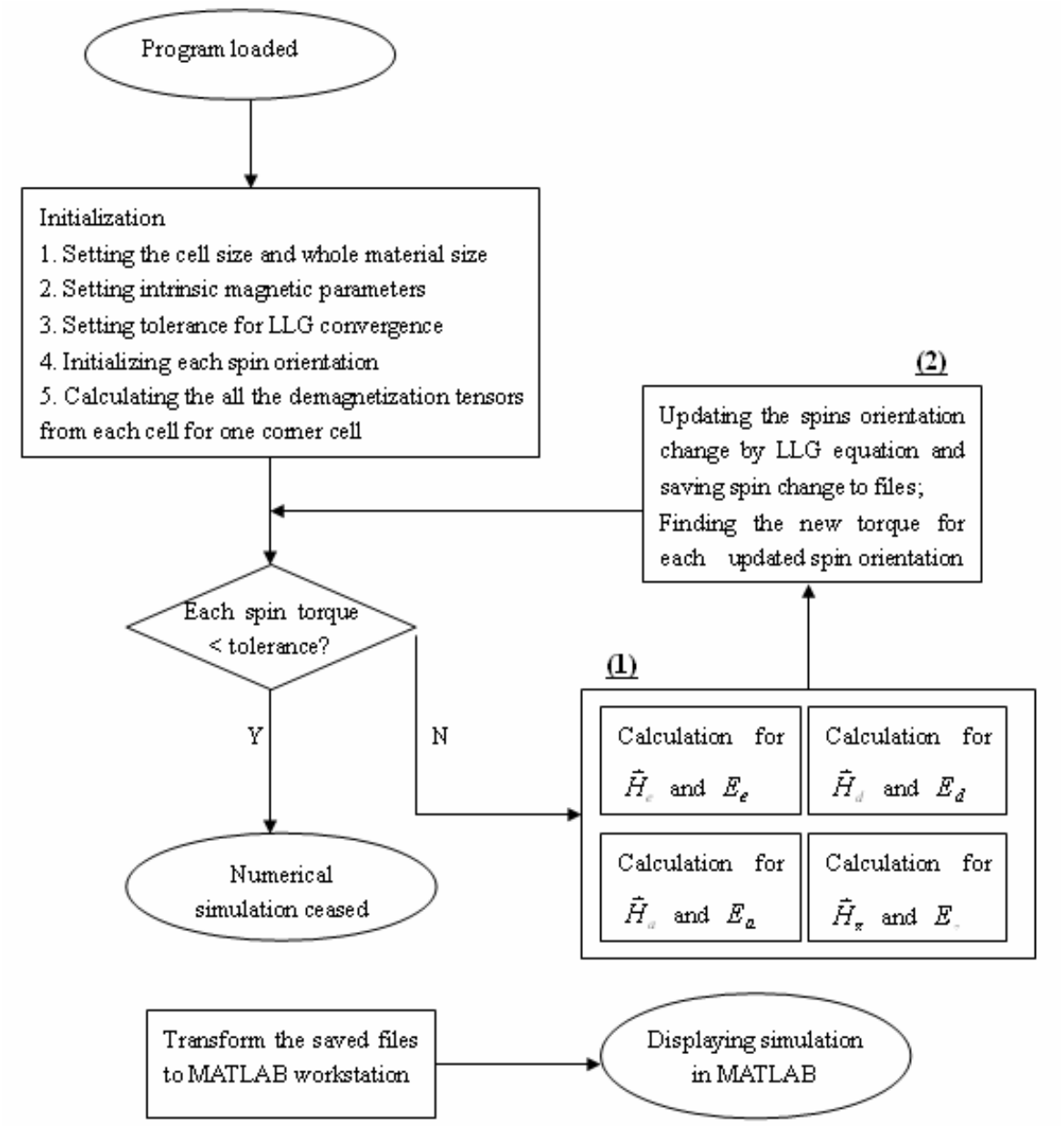


Figure 6. Micromagnetics software Flow Chart

The codes for the four modules part **(1)** in *Figure 6* are attached in *Appendix 1* and the codes for part **(2)** are in *Appendix 2*.

3.2 Micromagnetics Program verification

3.2.1 Verification of Problem #3

The mumag (or micromagnetics) society has developed several "standard problems" to ensure that micromagnetics programs from various researchers produce consistent and correct results. This work is verified against mumag Standard Problem #3 which is available at http://www.ctcms.nist.gov/~rdm/mumag.org.html. This problem involves calculating the stable magnetic domains that occur in a cubic sample as a function of the size of the cube with given magnetic parameters. The sample parameters are summarized as the following,

K_1 (J/m^3)	A (J/m)	damping constant α	M_s	ensemble size (nm)	cell size (nm)
6.28315e4	1.7735e-12	0.75	10e6	$45\times45\times45$	$4.5\times4.5\times4.5$

Table 1. Sample Parameters set in Problem 3

Previous researchers have shown that when the edge length of the cube is large a closure domain or vortex state is the stable state. This state is shown below in Figure 11. As the edge length is reduced at some point the exchange energy dominates and the flower state becomes the stable state. The flower state has all of the spins aligned along the easy axis except at the opposite edges of the cube where the demag field forces the spins to point slightly outward, towards the corners of the cube. This transition is called the single domain limit. The single domain limit is the length of the sample at which the vortex and flower states have exactly the same energy. Previous researchers have shown that when the $sample\ size \approx 8\sqrt{A/K_m}$, the system holds the same energy for vortex and flower states. This work tested problem #3 starting from the parallel state and finally transformed to the flower state. Our code gives the estimate of the critical edge length as $\frac{cell\ size}{\sqrt{A/K_m}} \approx 8.47$

It falls in the range reported by other groups, 8.47 (Hubert, Favian and Rave), 8.4687 (Ribeiro, Paulo Freitas, and José Luís Martins) and 8.52 (Hertel and Kronmuller).

Result of this work in intrinsic units:

	Result of this work	Rave	Hertel	Martins
E	0.29094	0.3027	0.3027	0.3026
Ea	0.05561	0.0783	0.0830	0.0521
Ee	0.163506	0 .1723	0.1696	0.1724
Ed	0.071825	0.0521	0.0830	0.0780

Table 2. Result comparison for Problem 3

Our results are consistent with those previously reported and the figure shown in MATLAB is consistent with that reported by other researchers. *Figure 7* is the initial state and *Figure 8* is the final state with the torque is less than 10^{-5} for any spin in 2D.

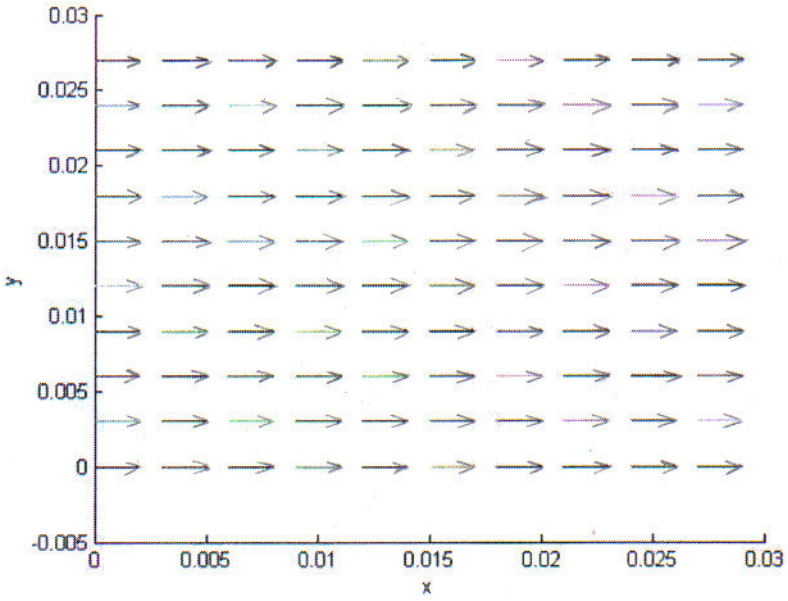


Figure 7. Initial state for Problem 3

The *figure 8* is a 2D display which shows the top of this cubic sample of *Figure 9.*

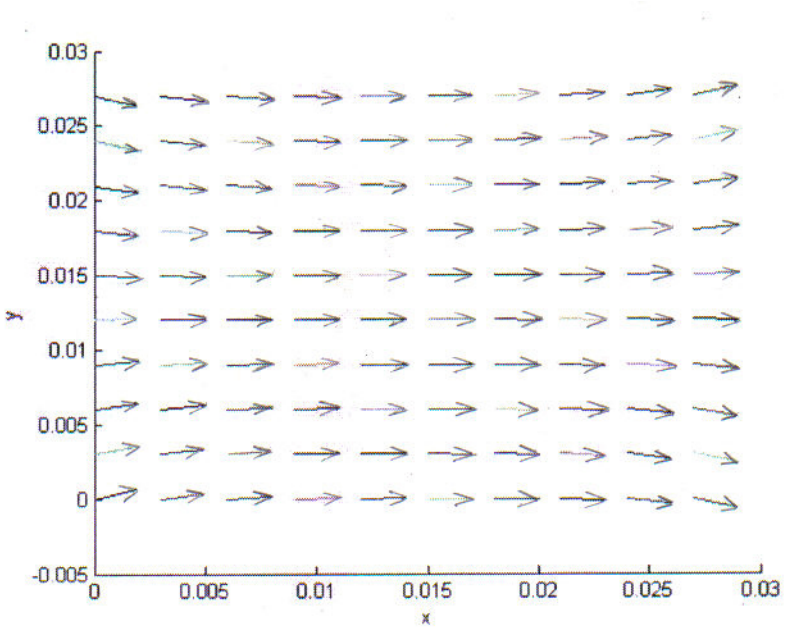


Figure 8. Final state for Problem 3

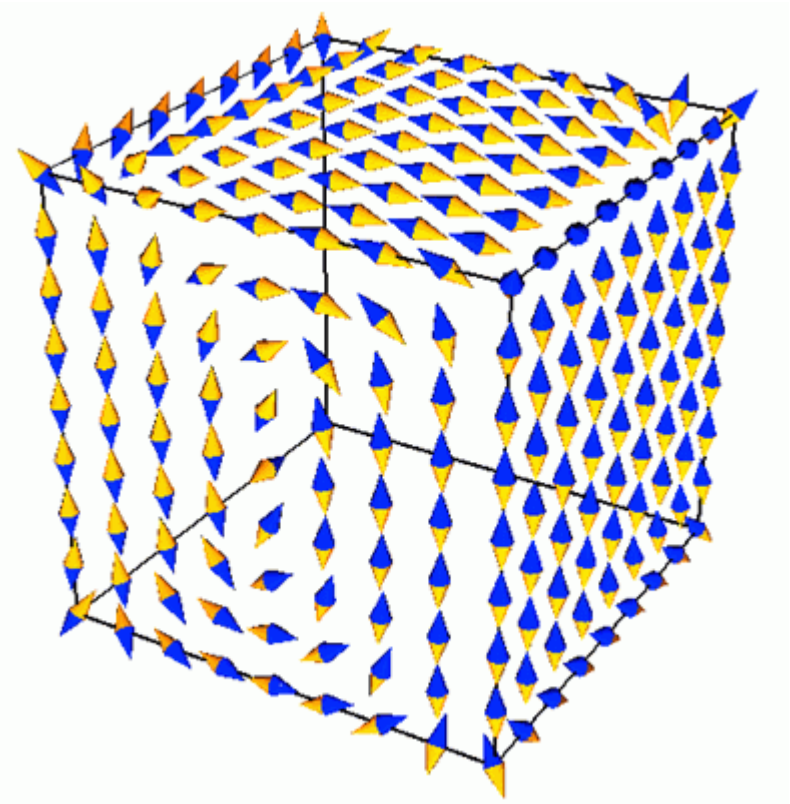


Figure 9 Vortex State in 3D display

(cited from http://www.ctcms.nist.gov/~rdm/mumag.org.html)

Chapter 4 Application

This work can easily set the spins to some state, while this is difficult for OOMMF, which stands for Object Oriented Micromagnetics Software and is available from http://math.nist/gov/oommf .

4.1 One Example

The initial configuration is in the state of either “head-to-head” or “tail-to-tail” each other one by one, refer to the *Figure 10*. OOMMF supplies several general initial states however it does not supply this starting point.

The sample parameters are summarized as the following,

K_1 (J/m^3)	A (J/m)	damping constant α	M_s	ensemble size (nm)	cell size (nm)
500	13e-12	0.5	8e5	$200\times200\times20$	$20\times20\times20$

Table 3. Sample Parameters set in Problem 3

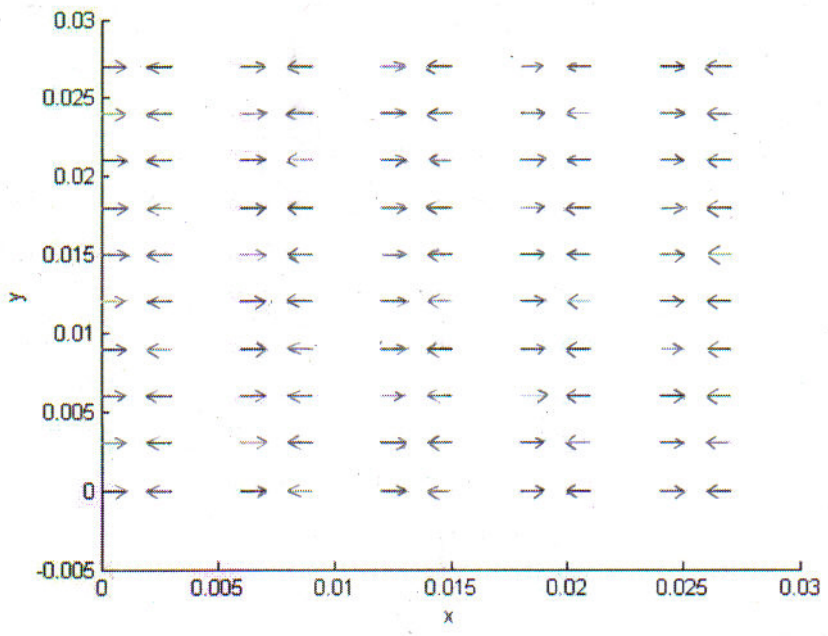


Figure 10. Initial state for an example of 10×10 cells

The configuration parameters are set as *Table 3*. *Figure 11* shows how to set up these values in a head file of this C++ program.

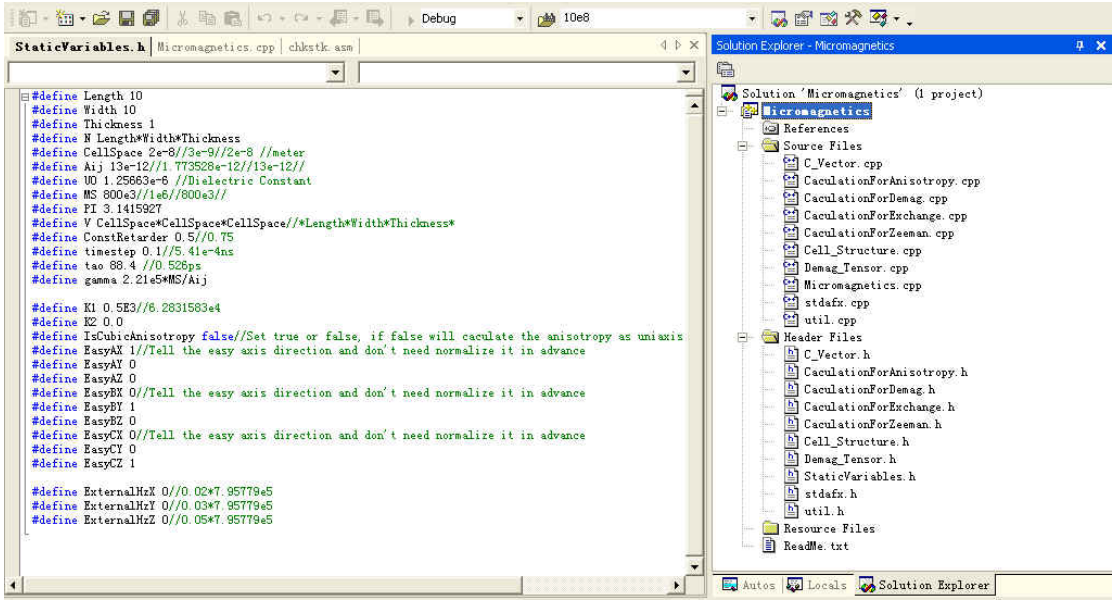


Figure 11. Setting for the head file "StaticVariables.h" in this work

Figure 12 gives the final result for this example which shows that the sample forms a vortex state. The various energy terms in units of J/m^3 terms are given in *Table 4*.

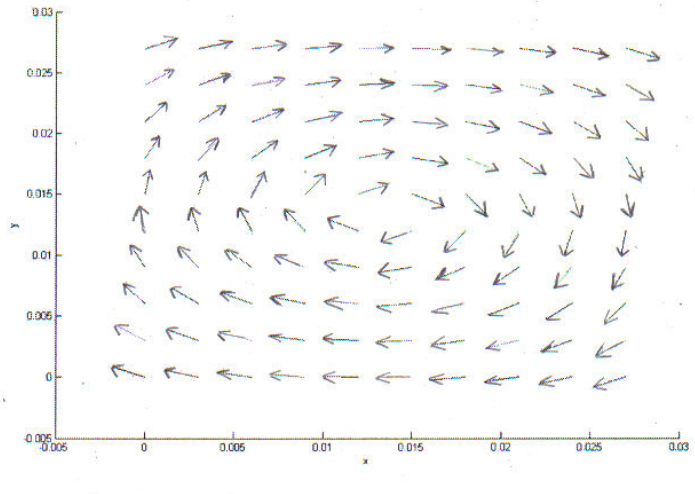


Figure 12. Final state for an example of 10×10 cells

	Initial State	Final State
E	441161	28754
Ea	0	9920
Ed	441161	14098.8
Ee	0	4735

Table 4. Result for final energy terms

From *Figure 11*, the final state is a vortex which means the demagnetization energy reaches a minimum point. The decrease in the demagnetization energy is offset by an increase in the exchange energy from 0 to 4735 J/m^3 since all the spins are not parallel to each other and not all of the spins lie along the easy axis (1, 0, 0) resulting in a non-zero value for the anisotropy energy from 0 to 9920 J/m^3.

The total energy is also decreased from 441161 to 28754 J/m^3 with the maximum torque is only 9.8075×10^{-6}. This value of the torque suggests that the ensemble reaches a metastable state.

Chapter 5 Conclusions

A micromagnetics modeling program for use on a single computer has been successfully developed in C++. This simulation results for this program have been verified by comparison to mumag Standard Problem #3. Such a modeling program can be used for simulating the domain patterns and reversal process of magnetic materials. This program can simulate a 3D ensemble of spins up to 10^5 cells based upon a single workstation with 512M of memory.

This work can also provide a visualization of the magnetic domain information inside the material using a module in the program which allows the program to use MATLAB for 3D display. Finally, this program allows the user to easily set the initial state including initial orientation of the spins.

Future work should include modification of the code so that it can easily take advantage of software platforms for parallel workstation, such CACTUS which was designed for scientists and engineers [23]. CACTUS has a central core code which connects to the application modules (thorns) through a standard interface. Therefore, the program presented here can be recompiled as a thorn for micromagnetics calculation and then be easily ported to a parallel architecture. By assuming such an architecture the program can share computer resource, such as additional processors and memory, with a CACTUS server. With sufficient resources the program can be used to model very large magnetic systems, such as the magnetic field inside and around the Earth and can provide a dynamic model of field reversing of the Earth's magnetic field.

Reference

[1] Wu, Zi-mu, "Meng Liang Lu", A.D.1274

[2] David P. Stern, Reviews of Geophysics, 27, 103-114, 1989.

[3] William F. Brown, Jr., Phys. Rev. 130, 677 1963

[4] William Fuller Brown, "Micromagnetics" Interscience Publishers 1963

[5] M.R. Scheinfein, J. Unguris, J.L. Blue, *Phys. Rev. B.* **43**(4), 3395-3422 (1991)

[6] Helmut Kronmuller and Manfred Fahnle, "Micromagnetism and the Microstructure of Ferromagnetic Solids"

[7] Scott L. Whittenburg, Micromagnetics of Nanoparticles Chap 11 425-439

[8] Hanning Chen and Scott L. Whittenburg, J. App. Phys. 94 8 5278-5282 2003

[9] Zhizhong Zhao Graduation Dissertation, UNO, Appendix A 131 2004

[10] Dennis Schabes and Byron Lengsfield, J. App. Phys. 95 6 3175-3201 2004

[11] Josef Fidler and Thomas Schrefl, J. App. Phys. 33 R135-R156 2000

[12] Bruce A. Buffett Science 288 16 JUNE 2007-2012 2000

[13] A. E. Labonte J. Appl. Phys. 40 2450-2458 1969

[14] V. Tsiantos, T. Schrefl, W. Scholz, H. Forster, D. Suess http://magnet.atp.tuwien.ac.at/scholz/projects/papers/preprint/tsiantos/emsa02/emsa2002.pdf

[15] W. F. Brown, Magnetostatic Principles in Ferromagnetism, North-Holland Pub. Inc., New York 1962

[16] A. D. Becke Phys. Rev. A 38 6 3098-3100 1988

[17] H. Brooks, Phys. Rev. 58, 909 1940

[18] G. Bertotti, Hysteresis in Magnetism

[19] L. Greengard and V. Rokhlin, J. Comp. Phys. 73, 325-348 1987

[20] J. Blue and M. Scheinfei, IEEE Trans. Magn. 27, 4778 1991

[21] S. Yuan and H. N. Bertram, IEEE Trans. Magn. 28. 2031 1992

[22] A. J. Newell, W. Williams and D. J. Dunlop, J. of Geo. Res. 98, 9551-9556 1993

[23] Gabrielle Allen, Tom Goodale, Cactus Team, http://www.cactuscode.org/

Appendices

Appendix A Calculation Algorithms for four fields

Appendix A.1 Exchange Field

```
double ExchangeEnergyCaculator(Cell_Structure cell1,Cell_Structure cell2, double A)
{
    double temp;
    C_Vector tempMagnetization;
    tempMagnetization.x=cell2.M.x-cell1.M.x;
    tempMagnetization.y=cell2.M.y-cell1.M.y;
    tempMagnetization.z=cell2.M.z-cell1.M.z;
    temp=0.25*A*Norm(tempMagnetization)*Norm(tempMagnetization);
    return(temp);
}
C_Vector ExchangeFieldCaculator(Cell_Structure cell1,Cell_Structure cell2, double A)
{
    C_Vector temp;
    C_Vector tempMagnetization;
    tempMagnetization.x=cell2.M.x-cell1.M.x;
    tempMagnetization.y=cell2.M.y-cell1.M.y;
    tempMagnetization.z=cell2.M.z-cell1.M.z;
    temp.x=A*tempMagnetization.x;
    temp.y=A*tempMagnetization.y;
```

```
    temp.z=A*tempMagnetization.z;
    return(temp);}
void CaculationForExchange(Cell_Structure Cell[N],double ExchangeConst1)
{
    C_Vector tempi;
    double tempEnergyDensity;
    for(int i=0;i<N;i++)
        {
        Cell[i].Ee=0;
        Cell[i].He=Initialization(Cell[i].He);
        }

    for(int i=0;i<N;i++)
        {   for(int j=i+1;j<N;j++)
            {
                if(Distance(Cell[i].R,Cell[j].R)<=1.5*CellSpace)
                {
                tempi=ExchangeFieldCaculator(Cell[i],Cell[j],ExchangeConst1);
                Cell[i].He.x=Cell[i].He.x+tempi.x;
                Cell[i].He.y=Cell[i].He.y+tempi.y;
                Cell[i].He.z=Cell[i].He.z+tempi.z;
                Cell[j].He.x=Cell[j].He.x-tempi.x;
                Cell[j].He.y=Cell[j].He.y-tempi.y;
                Cell[j].He.z=Cell[j].He.z-tempi.z;
```

```
tempEnergyDensity=ExchangeEnergyCaculator(Cell[i],Cell[j],ExchangeConst1);
                    Cell[i].Ee=Cell[i].Ee+tempEnergyDensity;
                    Cell[j].Ee=Cell[j].Ee+tempEnergyDensity;
                    }
                }
            }
```

Appendix A.2 Anisotropy Field

```
void CaculationForAnisotropy(Cell_Structure Cell[N])//Only for uniaxial case and K1>0;
{

    double k1,k2;
    double tempdotMa,tempdotMb,tempdotMc;
    double aCoefficient,bCoefficient,cCoefficient;
    double tempdot=0,tempdot1=0;
    for(int i=0;i<N;i++)
    {
        Cell[i].Ea=0;
        Cell[i].Ha=Initialization(Cell[i].Ha);
    }
    if(IsCubicAnisotropy)
    {   C_Vector a;       //the direction of easy axis
        a.x=EasyAX;
```

```
a.y=EasyAY;

a.z=EasyAZ;

a=normalize(a);

C_Vector b;       //the direction of easy axis

b.x=EasyBX;

b.y=EasyBY;

b.z=EasyBZ;

b=normalize(b);

C_Vector c;       //the direction of easy axis

c.x=EasyCX;

c.y=EasyCY;

c.z=EasyCZ;

c=normalize(c);

for(int i=0;i<N;i++)

{

    k1=GetConversionk1(Cell[i].Ms);

    k2=GetConversionk2(Cell[i].Ms);

    tempdotMa=ProductofDot(Cell[i].M,a);

    tempdotMb=ProductofDot(Cell[i].M,b);

    tempdotMc=ProductofDot(Cell[i].M,c);

Cell[i].Ea=k1/2*(tempdotMa*tempdotMa*tempdotMb*tempdotMb+tempdotMa*tempdotMa*tempdotMc
*tempdotMc+tempdotMc*tempdotMc*tempdotMb*tempdotMb)+k2/2*(tempdotMa*tempdotMb*tempdotMc*t
empdotMa*tempdotMb*tempdotMc);

aCoefficient=-(tempdotMb*tempdotMb+tempdotMc*tempdotMc)*k1-tempdotMb*tempdotMb*tempdotM
```

```
c*tempdotMc*k2;

    bCoefficient=-(tempdotMa*tempdotMa+tempdotMc*tempdotMc)*k1-tempdotMa*tempdotMa*tempdotM
c*tempdotMc*k2;

    cCoefficient=-(tempdotMa*tempdotMa+tempdotMb*tempdotMb)*k1-tempdotMa*tempdotMa*tempdotM
b*tempdotMb*k2;

                Cell[i].Ha.x=aCoefficient*a.x+bCoefficient*b.x+cCoefficient*c.x;

                Cell[i].Ha.y=aCoefficient*a.y+bCoefficient*b.y+cCoefficient*c.y;

                Cell[i].Ha.z=aCoefficient*a.z+bCoefficient*b.z+cCoefficient*c.z;

            }

        }

        else

        {    C_Vector a;       //the direction of easy axis

            a.x=EasyAX;

            a.y=EasyAY;

            a.z=EasyAZ;

            a=normalize(a);

        if(K1>=0)

        {

            for(int i=0;i<N;i++)

            {

            k1=GetConversionk1(Cell[i].Ms);

            k2=GetConversionk2(Cell[i].Ms);

            tempdot1= ProductofDot(Cell[i].M,a);

            tempdot=tempdot1*tempdot1;
```

```
        Cell[i].Ea=0.5*(k1*(1-tempdot)+k2*(1-tempdot)*(1-tempdot));


        Cell[i].Ha.x=(k1+2*k2*(1-tempdot))*tempdot1*a.x;
        Cell[i].Ha.y=(k1+2*k2*(1-tempdot))*tempdot1*a.y;
        Cell[i].Ha.z=(k1+2*k2*(1-tempdot))*tempdot1*a.z;
        }
}
else
{
        for(int i=0;i<N;i++)
        {
        k1=GetConversionk1(Cell[i].Ms);
        k2=GetConversionk2(Cell[i].Ms);
        tempdot1= ProductofDot(Cell[i].M,a);
        tempdot=tempdot1*tempdot1;


        if(k1<0)Cell[i].Ea=0.5*(-k1*tempdot+k2*(1-tempdot)*(1-tempdot));
        else Cell[i].Ea=0.5*(k1*(1-tempdot)+k2*(1-tempdot)*(1-tempdot));


        Cell[i].Ha.x=(k1+2*k2*(1-tempdot))*tempdot1*a.x;
        Cell[i].Ha.y=(k1+2*k2*(1-tempdot))*tempdot1*a.y;
        Cell[i].Ha.z=(k1+2*k2*(1-tempdot))*tempdot1*a.z;
        }
```

```
        }

        }

}
```

Appendix A.3 Demagnetization Field

```
void CaculationForDemag(Cell_Structure Cell[N],Demag_Tensor
TensorFromCode[Thickness][Width][Length])

{

        for(int i=0;i<N;i++)

        {

                Cell[i].Ed=0;

                Cell[i].Hd.x=0;

                Cell[i].Hd.y=0;

                Cell[i].Hd.z=0;

        }

        C_Vector temph;



        for(int i=0;i<N;i++)//Caculation for the demag from other cells

        {

                for(int j=i+1;j<N;j++)

                {

                double xx=(Cell[j].R.x-Cell[i].R.x)/CellSpace, yy=(Cell[j].R.y-Cell[i].R.y)/CellSpace,
zz=(Cell[j].R.z-Cell[i].R.z)/CellSpace;



                Demag_Tensor tensor = TensorFromCode[int(abs(zz)+0.1)][int(abs(yy)+0.1)][int(abs(xx)+0.1)];

                double NXX=tensor.Nxx, NXY=tensor.Nxy, NXZ=tensor.Nxz;
```

```
double NYY=tensor.Nyy, NYZ=tensor.Nyz, NZZ=tensor.Nzz;

if (xx<0) { NXY=-NXY; NXZ=-NXZ; }

if (yy<0) { NXY=-NXY; NYZ=-NYZ; }

if (zz<0) { NXZ=-NXZ; NYZ=-NYZ; }

temph.x=-Cell[j].Ms/Cell[i].Ms*(NXX*Cell[j].M.x+NXY*Cell[j].M.y+NXZ*Cell[j].M.z);

temph.y=-(NXY*Cell[j].M.x+NYY*Cell[j].M.y+NYZ*Cell[j].M.z);

temph.z=-(NXZ*Cell[j].M.x+NYZ*Cell[j].M.y+NZZ*Cell[j].M.z);

Cell[i].Hd.x=Cell[i].Hd.x+temph.x;

Cell[i].Hd.y=Cell[i].Hd.y+temph.y;

Cell[i].Hd.z=Cell[i].Hd.z+temph.z;

Cell[i].Ed=Cell[i].Ed-0.5*ProductofDot(Cell[i].M,temph);//4.22e5*Cell[j].Ms/Cell[i].Ms*ProductofDot(C
ell[i].M,temph);

temph.x=-(NXX*Cell[i].M.x+NXY*Cell[i].M.y+NXZ*Cell[i].M.z);

temph.y=-(NXY*Cell[i].M.x+NYY*Cell[i].M.y+NYZ*Cell[i].M.z);

temph.z=-(NXZ*Cell[i].M.x+NYZ*Cell[i].M.y+NZZ*Cell[i].M.z);

Cell[j].Hd.x=Cell[j].Hd.x+temph.x;

Cell[j].Hd.y=Cell[j].Hd.y+temph.y;

Cell[j].Hd.z=Cell[j].Hd.z+temph.z;
```

```
        Cell[j].Ed=Cell[j].Ed-0.5*ProductofDot(Cell[j].M,temph);//4.22e5*Cell[i].Ms/Cell[j].Ms*ProductofDot(Cell[j].M,temph);

            }

        }


        for(int i=0;i<N;i++)//Caculation for self-demag

            {

            double NXX=0.333333333333333, NXY=0, NXZ=0;

            double NYY=0.333333333333333, NYZ=0, NZZ=0.333333333333333;


            temph.x=-(NXX*Cell[i].M.x+NXY*Cell[i].M.y+NXZ*Cell[i].M.z);

            temph.y=-(NXY*Cell[i].M.x+NYY*Cell[i].M.y+NYZ*Cell[i].M.z);

            temph.z=-(NXZ*Cell[i].M.x+NYZ*Cell[i].M.y+NZZ*Cell[i].M.z);


            Cell[i].Hd.x=Cell[i].Hd.x+temph.x;

            Cell[i].Hd.y=Cell[i].Hd.y+temph.y;

            Cell[i].Hd.z=Cell[i].Hd.z+temph.z;

            Cell[i].Ed=Cell[i].Ed-0.5*ProductofDot(Cell[i].M,temph);//4.22e5*ProductofDot(Cell[i].M,temph);


            }

}
```

Appendix A.4 Zeeman Field

```
void CaculationForZeeman(Cell_Structure Cell[N])

{

    C_Vector ExternalHz;
```

```
    ExternalHz.x=ExternalHzX;

    ExternalHz.y=ExternalHzY;

    ExternalHz.z=ExternalHzZ;



    for(int i=0;i<N;i++)

    {

        Cell[i].Hz.x=ExternalHz.x/Cell[i].Ms;

        Cell[i].Hz.y=ExternalHz.y/Cell[i].Ms;

        Cell[i].Hz.z=ExternalHz.z/Cell[i].Ms;


        Cell[i].Ez=-(Cell[i].Hz.x*Cell[i].M.x+Cell[i].Hz.y*Cell[i].M.y+Cell[i].Hz.z*Cell[
    i].M.z);

    }

}
```

Appendix B Calculation Algorithms for LLG equation

```
C_Vector k;

C_Vector CrossProduct1;

C_Vector CrossProduct2;

double maxtorque;


FILE *fp2D=fopen( "mydata_2d.dat", "w" );

FILE *fp3D=fopen( "mydata_3d.dat", "w" );

FILE *fpE=fopen( "energy.dat", "w" );


do{

    maxtorque = 0;

    CaculationForExchange(Cell,ExchangeConst);

    CaculationForAnisotropy(Cell);

    CaculationForDemag(Cell,DemagTensor);

    CaculationForZeeman(Cell);


    for(int i=0;i<N;i++)

    {

        Cell[i].Etotal=Cell[i].Ee+Cell[i].Ea+Cell[i].Ed+Cell[i].Ez;

        Cell[i].Htotal.x=Cell[i].He.x+Cell[i].Ha.x+Cell[i].Hd.x+Cell[i].Hz.x;

        Cell[i].Htotal.y=Cell[i].He.y+Cell[i].Ha.y+Cell[i].Hd.y+Cell[i].Hz.y;
```

```
            Cell[i].Htotal.z=Cell[i].He.z+Cell[i].Ha.z+Cell[i].Hd.z+Cell[i].Hz.z;

        }


        for(int i=0;i<N;i++)

        {

            CrossProduct1=ProductofCross(Cell[i].M,Cell[i].Htotal);

            Cell[i].torque=Norm(CrossProduct1);

            if(maxtorque<Cell[i].torque) maxtorque=Cell[i].torque;



            CrossProduct2=ProductofCross(Cell[i].M,CrossProduct1);

            k.x=-CrossProduct1.x-ConstRetarder*CrossProduct2.x;

            k.y=-CrossProduct1.y-ConstRetarder*CrossProduct2.y;

            k.z=-CrossProduct1.z-ConstRetarder*CrossProduct2.z;



            //The adaptive timestep should be set up here, need further work

            Cell[i].M.x=Cell[i].M.x+k.x*timestep;

            Cell[i].M.y=Cell[i].M.y+k.y*timestep;

            Cell[i].M.z=Cell[i].M.z+k.z*timestep;

            Cell[i].M=normalize(Cell[i].M);

            if(ntime%5000==0)//Write a data into the recording file every 5000 iteratives

            {

                fprintf(fp3D, " %e %e %e %e %e %e\n",
Cell[i].R.x*MS,Cell[i].R.y*MS,Cell[i].R.z*MS,Cell[i].M.x,Cell[i].M.y,Cell[i].M.z);

                fprintf(fp2D, " %e %e %e %e\n", Cell[i].R.x*MS,Cell[i].R.y*MS,Cell[i].M.x,Cell[i].M.y);
            }
```

```
        printf("Cell[%d] M is(%g %g %g)\n",i,Cell[i].M.x,Cell[i].M.y,Cell[i].M.z);
        }
        if(ntime%1000==0)//Write a data into the recording file every 1000 iteratives
            {
                sumTotalE=0;
                for(int i=0;i<N;i++)  {sumTotalE=sumTotalE+Cell[i].Etotal;}
                ii++;fprintf(fpE, " %d %e\n",ii, sumTotalE*(1./1000)*U0*MS*MS);
            }
        printf("Present maxium torque is %e\n",maxtorque);
        printf("No. of Iteration is %e\n\n",time);
        time=time+timestep;
        ntime--;
}while(maxtorque>1e-5&&ntime>0);
```

Appendix C Fast Fourier Transform

```
long FFTLengthMax;

Complex * OmegaFFT;

Complex * ArrayFFT0, * ArrayFFT1;

Complex * ComplexCoef;

double FFTSquareWorstError;

long AllocatedMemory;


/*Initialization*/

void InitializeFFT(long MaxLength)

{

   long i;

   double Step;


   FFTLengthMax = MaxLength;

   OmegaFFT = (Complex *) malloc(MaxLength/2*sizeof(Complex));

   ArrayFFT0 = (Complex *) malloc(MaxLength*sizeof(Complex));

   ArrayFFT1 = (Complex *) malloc(MaxLength*sizeof(Complex));

   ComplexCoef = (Complex *) malloc(MaxLength*sizeof(Complex));

   Step = 2.*PI/(double) MaxLength;

   for (i=0; 2*i<MaxLength; i++) {

      OmegaFFT[i].R = cos(Step*(double)i);
```

```
        OmegaFFT[i].I = sin(Step*(double)i);
    }
    FFTSquareWorstError=0.;
    AllocatedMemory = 7*MaxLength*sizeof(Complex)/2;
}

/*do recursion of FFT*/
void RecursiveFFT(Complex * Coef, Complex * FFT, long Leng, long Step, long Sign)
{
    long i, OmegaStep;
    Complex * FFT0, * FFT1, * Omega;
    double tmpR, tmpI;

    if (Leng==2) {
        FFT[0].R = Coef[0].R + Coef[Step].R;
        FFT[0].I = Coef[0].I + Coef[Step].I;
        FFT[1].R = Coef[0].R - Coef[Step].R;
        FFT[1].I = Coef[0].I - Coef[Step].I;
        return;
    }

    FFT0 = FFT;
    RecursiveFFT(Coef,FFT0,Leng/2,Step*2,Sign);
    FFT1 = FFT+Leng/2;
```

```
    RecursiveFFT(Coef+Step,FFT1,Leng/2,Step*2,Sign);


    Omega = OmegaFFT;
    OmegaStep = FFTLengthMax/Leng;
    for (i=0; 2*i<Leng; i++, Omega += OmegaStep) {
      /* Recursion formula for FFT :
           FFT[i]                <-   FFT0[i] + Omega*FFT1[i]
           FFT[i+Length/2] <-   FFT0[i] - Omega*FFT1[i],
           Omega = exp(2*I*PI*i/Length) */
      tmpR = Omega[0].R*FFT1[i].R-Sign*Omega[0].I*FFT1[i].I;
      tmpI = Omega[0].R*FFT1[i].I+Sign*Omega[0].I*FFT1[i].R;
      FFT1[i].R = FFT0[i].R - tmpR;
      FFT1[i].I = FFT0[i].I - tmpI;
      FFT0[i].R = FFT0[i].R + tmpR;
      FFT0[i].I = FFT0[i].I + tmpI;
    }
}


/* Compute the complex Fourier Transform of Coef into FFT*/
void FFT(double *Coef,long Leng, Complex *FFT,long NFFT)
{
    long i;
    /* Transform array of real coefficient into array of complex    */
    for (i=0; i<Leng; i++) {
```

```
        ComplexCoef[i].R = Coef[i];
        ComplexCoef[i].I = 0.;
    }
    for (; i<NFFT; i++)
        ComplexCoef[i].R = ComplexCoef[i].I = 0.;
    RecursiveFFT(ComplexCoef,FFT,NFFT,1,1);
}

void InverseFFT(Complex * FFT, long NFFT, double * Coef, long Leng)
{
    long i;
    double invNFFT = 1./(double) NFFT, tmp;
    RecursiveFFT(FFT, ComplexCoef, NFFT, 1, -1);
    for (i=0; i<Leng; i++) {
        /* Closest integer to ComplexCoef[i].R/NFFT */
        tmp = invNFFT*ComplexCoef[i].R;
        Coef[i] = tmp;
        //if ((tmp-Coef[i])*(tmp-Coef[i])>FFTSquareWorstError)
            //FFTSquareWorstError = (tmp-Coef[i])*(tmp-Coef[i]);
    }
}
```

Appendix 4 Units for Magnetic Properties in cgs emu and SI system

	Symbol	Gauss & cgs emu[a]	Conversion Factor	SI & rationalized mks[c]
Magnetic flux density, magnetic induction	B	Gauss (G)[d]	10^{-4}	tesla (T), Wb/m^2
Magnetic flux	Φ	Maxwell (Mx), $G \cdot cm^2$	10^{-8}	weber (Wb), volt second (V·s)
Magnetic potential difference, magnetomotive force	U, F	Gilbert (Gb)	$10/4\pi$	ampere (A)
Magnetic field strength, magnetizing force	H	Oersted (Oe), [e] Gb/cm	$10^3/4\pi$	A/m[f]
(Volume) magnetization	M	emu/cm^3[h]	10^3	A/m
(Volume) magnetization	4πM	G	$10^3/4\pi$	A/m
Magnetic polarization, intensity of magnetization	J, I	emu/cm^3	$4\pi \times 10^{-4}$	T, Wb/m^2[i]
(Mass) magnetization	σ, M	emu/g	1 $4\pi \times 10^{-7}$	$A \cdot m^2/kg$ Wb·m/kg
Magnetic moment	m	emu, erg/G	10^{-3}	$A \cdot m^2$, joule per tesla (J/T)
(Volume) susceptibility	χ, κ	dimensionless, emu/cm^3	4π $(4\pi)2 \times 10^{-7}$	dimensionless henry per meter (H/m), Wb/(A·m)
(Mass) susceptibility	χ_ρ, κ_ρ	cm^3/g, emu/g	$4\pi \times 10^{-3}$ $(4\pi)2 \times 10^{-10}$	m^3/kg $H \cdot m^2/kg$
(Molar) susceptibility	χ_{mol}, κ_{mol}	cm^3/mol, emu/mol	$4\pi \times 10^{-6}$ $(4\pi)2 \times 10^{-13}$	m^3/mol $H \cdot m^2/mol$
Permeability	μ	dimensionless	$4\pi \times 10^{-7}$	H/m, Wb/(A·m)
Relative permeability	μ_r	not defined	—	dimensionless
(Volume) energy density, energy product	W	erg/cm^3	10^{-1}	J/m^3

From R. B. Goldfarbt, Department of Commerce, National Bureau of Standards.

Vita

The author was born in Beijing in 1980. He obtained his first bachelor degree in Polymer Chemistry in 2003 and dual bachelor degree in Computer Application in 2002 at University of Science and Technology of China. He was also coenrolled as a research assistant in Technical Institute of Physical and Chemistry, Chinese Academy of Sciences from 2002 to 2003. Then he joined the chemistry department of University of New Orleans in spring 2004 and became a member of Professor Scott Whittenburg's research group.