A Study of Switching in Magnetic Tunneling Junctions

by

Hür Köşer

B.S., Massachusetts Institute of Technology (1998)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

February 1999

Signature of Author .....
Department of Electrical Engineering and Computer Science
31 January 1999

Certified by .....
Terry P. Orlando
Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairperson, Department Committee on Graduate Students

A Study of Switching in Magnetic Tunneling Junctions

by

Hür Köşer

Submitted to the Department of Electrical Engineering and Computer Science
on 31 January 1999, in partial fulfillment of the
requirements for the degree of
Master of Engineering

Abstract

The dynamics of magnetization reversal in magnetic tunneling junctions are investigated both numerically and experimentally. A practical, large-scale micromagnetics simulation program is developed and used to study switching in these devices. Also, an experiment is setup to conduct switching characterization in magnetic tunneling junctions with short width pulses. In both the simulation and the experiment, it is observed that the switching dynamics are dominated by the degree of damping inside the magnetic medium of the top electrode. The findings from these studies suggest a series of engineering design points that are relevant for the development and implementation of a magnetic random access memory (MRAM).

Thesis Supervisor: Terry P. Orlando
Title: Professor of Electrical Engineering

Thesis Supervisor: Dr. William Gallagher
Title: Manager, MRAM Group

Contents

List of Figures

Chapter 1

Introduction

The increasing demands of the computer market for faster, denser information storage have determined the trend in solid-state memory development for the past several decades. The shrinkage in size of CMOS technology has been relatively rapid, corresponding to a doubling of device density for every eighteen months since 1975 (see Figure 1-1). At the same time, computer processors have been doubling their speed every two years.

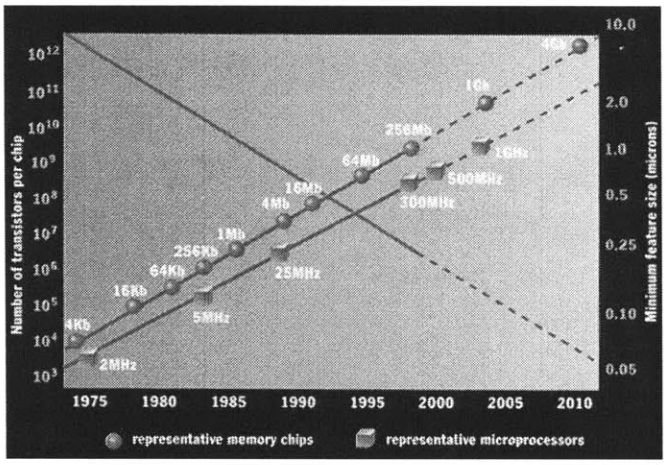

Figure 1-1: The development of computer memory size and processor speed since 1975. Dashed lines are projections for the future. (Source: IBM Research, Number 3, 1998)

Today, computation power and speed is more dependent on memory size and access time than the microprocessor clock rate. Dynamic Random Access Memory (DRAM) chips are

still the most commonly used RAM units. Unfortunately, they are slow compared to the computer processors they are used with. The bits stored in a DRAM need to be refreshed often during operation, and the information is lost at power down. These considerations have motivated research for alternative high-density storage devices. It is in this perspective that magnetoresistive devices have received attention in recent years.

Magnetoresistance (MR) is the change in a material's resistivity as a result of an applied magnetic field. The effect is more pronounced in magnetic metals. One such phenomenon, called the anisotropic magnetoresistance (AMR) effect, refers to the dependence of the resistivity of certain magnetic metals to the relative directions of applied current and magnetization. The change in resistivity is relatively small (5% for permalloy); nevertheless, high capacity hard disk drives that successfully utilize the AMR effect are already commercially available.

Another MR effect, giant magnetoresistance (GMR), offers a larger resistance change – as high as 25% in modest magnetic fields ($\lesssim$ 100 Oe) – compared to the AMR effect. The underlying structure in a GMR device is the presence of an interface between a magnetic and a nonmagnetic metal. A Fe/Cu/NiFe stack is one example of the multilayer devices that have been studied. It is believed that the underlying mechanism for the GMR effect includes two current channels of opposite spin directions, and the conductivity of each channel depends on the spin-dependent scattering at the interface. As an external field is applied, internal magnetic layers are aligned, and the larger of the two current channels contributes to the total conduction, reducing the resistance of the device. The most common memory application of the GMR effect is the sandwich multilayer (magnetic-nonmagnetic-magnetic), also known as a spin-valve. Spin-valves, with MR values of about 10%, are already appearing in the next generation hard disk technology to replace AMR memories.

Recently, a magnetoresistance effect discovered in manganese perovskites has attracted much attention, thanks to its huge value (>10000%). The effect has been rightfully named as Colossal Magnetoresistance (CMR). However, a magnetic field on the order of a few Teslas is required to observe the switching in resistance, making CMR devices somewhat impractical for immediate applications.

Particularly of interest for applications are magnetic tunnel junctions (MTJs). These devices are sandwich structures of two ferromagnetic metals separated by a thin insulating layer. The

resistivity of MTJs depends on the spin-dependent tunneling probability of electrons through the insulating barrier. The MR values in these devices are on the order of 30-50%, and the switching field is much less than those required for observing the CMR effects. Moreover, electron tunneling for any appreciable barrier thickness results in a high resistance, which means less power consumption for a given applied voltage. Also, MTJs are less prone to changes in temperature. All these properties make this class of devices a good candidate for practical applications.

1.1 Background

One of the earliest and most convincing successes of the theory of quantum mechanics is the explanation of tunneling phenomena. Classical rules of motion prohibit a particle from penetrating a potential barrier whose magnitude is larger than the total energy of the particle itself. In quantum mechanics, tunneling phenomenon arises naturally following the wave nature of matter. Alpha decay from radioactive nuclei produced the first strong evidence that confirmed quantum tunneling.

Since the formulation of quantum mechanics, numerous experiments have been devised to study tunneling in a variety of media. With the advent of photolithography techniques, it has been possible to manufacture thin enough insulator layers to observe tunneling phenomena in solid-state circuits as well[1]. Simmons ([7],[8]) formulated a general theory of electron tunneling between any two metals, and studied the effects of temperature on the I-V curves of the tunneling devices. Among many experimental reports of electron tunneling in metal-insulator-metal structures was that of Knorr[4], who devised an ellipsometrical method to produce insulator layers that reproducibly lied within a 20-30 Å thickness range. In this study, Knorr used aluminum as the bottom metal, and oxidized its surface in oxygen plasma to create the insulator aluminum oxide.

In 1975, Julliere[1] reported his study of tunneling between ferromagnetic films at liquid helium temperature (~4.2 K). His experimental devices were composed of iron and cobalt

[1]Quantum tunneling depends roughly exponentially on the length scale of the potential barrier. Hence, electron tunneling in a metal-insulator-metal structure can only be observed and studied appreciably with an insulating layer that is less than several hundred Angstroms thick.

electrodes, separated by a thin (100-150 Å) germanium film. It turned out that the apparent conductance of the junctions depended on the relative orientation of the magnetizations of the two ferromagnetic films. In particular, the conductance of a device was at a maximum when the two magnetizations were aligned parallel, and a minimum when they were anti-parallel. This meant that such devices could support two distinct states, implying the possibility of using these junctions in computer memory applications.

Numerous other experimental attempts followed Julliere's first report. In their 1982 paper, Maekawa and Gäfvert considered a simple model for electron tunneling in ferromagnet-insulator-ferromagnet (FM-I-FM) structures[10]. According to this model, the tunneling conductance depends on the density of states of the majority and minority spin electrodes of the ferromagnetic metals at the Fermi surface. The MR value, defined in their paper as the change in conductance divided by the average conductance, is simply $\Delta G/G \equiv P_A P_B$, where P_A and P_B denote the polarization of the tunneling density of states in the metals A and B respectively. According to studies conducted by Tedrow and Meservey with superconductor-insulator-ferromagnetic structures[5], the spin polarization factors for transition metals fall into the 20-40 % range. The corresponding MR values predicted by the model would then fall in the range of 8-32 %. However, in their study of Ni-NiO-Ni junctions, Maekawa and Gäfvert observed only a 0.5-1% effect.

Miyazaki[11], Suezawa[13], Yaoi[12] also reported MR values on the order of a percent or so. In most of these studies, the greatest challenge has been to process the MTJs with a uniform insulating layer without any pin-hole shorting. It is only recently that MR values in the expected range have been produced successfully.

Interest in magnetoresistive devices was suddenly renewed in 1995 when Miyazaki and Tezuka[14] reported an MR ratio of 30% at 4.2 K and 18% at room temperature. However, their results came from a single $Fe/Al_2O_3/Fe$ junction; other junctions prepared similarly had MR values in the 1-6% range. Fortunately, Moodera et al.[15] soon reported observing relatively high MR ratios (about 12%) reproducibly in their $Fe/Al_2O_3/FeCo$ junctions.

1.2 MTJs as memory elements

Our research group focuses almost entirely on the application potential of magnetic tunnel junctions[2]. The initial challenge has been to understand and control the parameters involved in manufacturing the MTJs - parameters such as the insulator thickness or the ferromagnetic electrode material. Processing has gone hand-in-hand with physical characterization of the devices produced, such as switching fields, and dependence on bias voltage and temperature ([18]-[20]). We now know, for instance, that application of several hundred millivolts across a junction will lower its MR value by half. This result implies certain limits on the voltages used by the sensing circuitry of a possible magnetic random access memory (MRAM).

On the engineering side, there are processing challenges of manufacturing sub-micron scale junctions, as well as integrating MTJs with complex CMOS circuitry. Also very important is the determination of the relationship between junction resistance and the processing variables. On the physics side, one needs to understand how different materials effect the MR ratio, what determines the switching field threshold and how fast MTJ devices can switch.

Previous research by other groups has been mostly done on junctions on the order of a millimeter square in area. For practical memory applications, however, this area needs to be taken below a micron square. Our group has already reported the successful fabrication of MTJs at tenth-micron dimensions by electron beam lithography[17]. The MR ratios at room temperature and in fields of a few tens of Oe have been reported to be 15-22% [16]. Most recently, more than 30% magnetoresistance has been achieved, and shown to be highly reproducible.

1.3 Fabrication

In this section, we will very briefly introduce the processing techniques used. For more detail, the interested reader is referred to the literature that describes the fabrication processes of magnetic tunneling junctions ([10]-[20]).

Figure 1-2 shows some of the steps involved in producing the MTJs that we have studied.

[2]See Reference [23] for possible implementations of MTJs as memory elements. References [22-23] consider certain design issues involved in designing a magnetoresistive memory element.

First, one starts by depositing the base electrode, the tunneling barrier and the top electrode of the junctions uniformly over a Si wafer. The base electrode usually consists of an antiferromagnetic layer and a ferromagnetic layer; the former provides an exchange bias for the latter, so that the bottom ferromagnetic layer is "pinned" down in one magnetization state. The barrier is usually formed by oxidizing a very thin metallic layer, either through O_2 plasma, or by long-term exposure to air. The top electrode (called the "free layer") is also ferromagnetic material (such as permalloy) that displays mainly a uniaxial magnetic anisotropy.

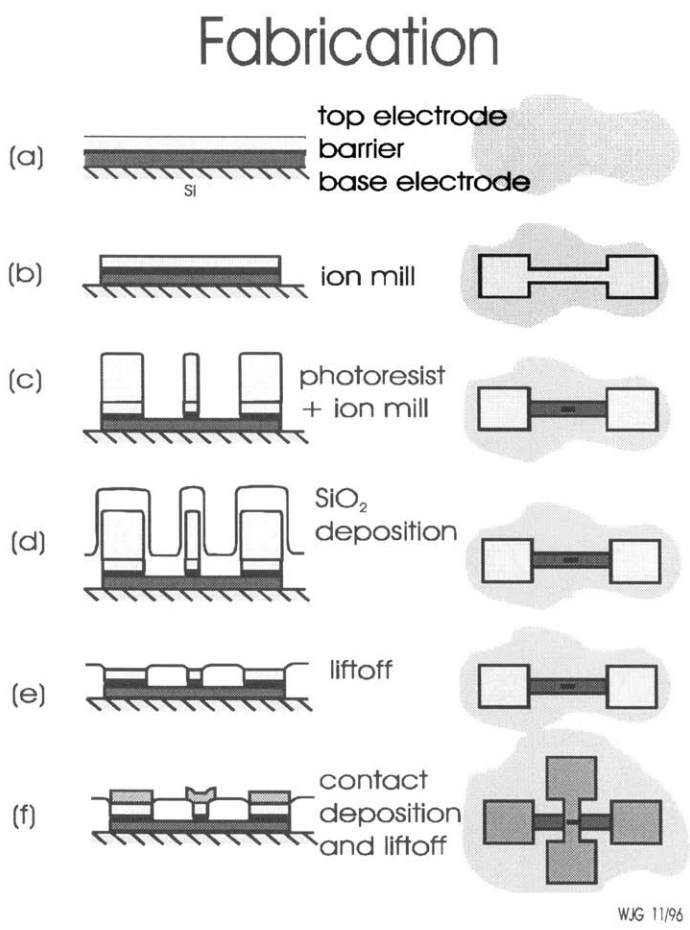

Figure 1-2: Some of the fabrication steps of the magnetic tunneling junctions and the associated electrical contacts.

Later, with ion milling, the bottom contact electrode shapes are etched. Photoresist is deposited on the remaining top electrode sites, followed by the deposition of an insulator (SiO_2)

that covers the exposed sections of the bottom electrode. Once the photoresist is lifted off, gold contact pads can be deposited. These pads provide the electrical contact to the top and the bottom electrodes of the MTJs, so that the resistance across the junctions can be measured.

Next generation samples are fabricated in about the same manner, except that the contacts to the bottom electrodes are now metallic vias, as opposed to very large tunneling junctions. For very small junction dimensions, electron beam lithograpy is used to define the devices.

1.4 Organization of this document

In the next chapter, we introduce the physical processes that determine magnetization dynamics in a ferromagnetic thin film. We will derive the Landau-Lifshitz-Gilbert equation that describes the time evolution of the total magnetic moment of an electrode. We will discuss our work on implementing this equation in a micromagnetics simulator, and present some our results from this simulation that raise issues relevant to the production of MRAM.

Chapter 3 begins with the introduction of our experimental work on magnetic tunneling junctions which are fabricated as described above. Our efforts are aimed at studying the characteristics of magnetization switching, especially with short width pulses, similar to a memory configuration. Once we describe the particular setup and the procedure, we will present our findings. The chapter will end with some remarks about the issues that come up from the results of the experiments.

In the last chapter, we summarize our findings from both the computer simulations and the experimental work. The significance of this study in understanding some of the dynamic switching characteristics of magnetic tunneling junctions will be pointed out. We conclude with a survey of the challenges still awaiting the researchers in the development of MRAM, and suggestions for future study will be made.

Appendix A briefly presents certain mathematical issues involved in integrating the Landau-Libshiftz-Gilbert equation numerically. Appendix B details the calculation of some geometrical factors that are used during the simulation. Finally, Appendix C documents the computer software code used during the micromagnetics simulations.

Bibliography

[1] M. Julliere, *Tunneling Between Ferromagnetic Films*, Phys. Lett., **54**, 225 (1975).

[2] W.K. Hiebert, A. Stankiewicz, M.R. Freeman, *Direct Observation of Magnetic Relaxation in a Small Permalloy Disk by Time-Resolved Scanning Kerr Microscopy*, Phys. Rev. Letts, **79**, 1134 (1997)

[3] J.C. Slonczewski, *Conductance and Exchange Coupling of Two Ferromagnets Separated by a Tunneling Barrier*, Phys. Rev. B, **39**, 6995 (1989)

[4] K. Knorr, J. D. Leslie, *Ellipsometrical Determination of Barrier Thicknesses of Metal-Insulator-Metal Tunnel Junctions*, Solid State Communications, **12**, 615 (1973)

[5] P. M. Tedrow and R. Meservey, *Spin Polarization of Electrons Tunneling from Films of Fe, Co, Ni, and Ni and its alloys*, Phys. Rev. B, **7**, 318 (1973)

[6] M. Mansuripur, *Magnetization Reversal Dynamics in the Media of Magneto-Optical Recording*, J. Appl. Phys., **63**, 5809 (1988)

[7] J. G. Simmons, *Electric Tunnel Effect between Dissimilar Electrodes Separated by a Thin Insulating Film*, J. Appl. Phys., **34**, 2581 (1963)

[8] J. G. Simmons, *Generalized Thermal J-V Characteristic for the Electric Tunnel Effect*, J. Appl. Phys., **35**, 2655 (1964)

[9] L. D. Landau, E. M. Lifshiftz and L. P. Pitaevski, *Electrodynamics of Continuous Media*, Pergamon Press, 1984

Experimental Reports

[10] S. Maekawa, U. Gäfvert, *Electron Tunneling Between Ferromagnetic Films*, IEEE Trans. Magn., **MAG-18**, 707 (1982)

[11] T. Miyazaki, T. Yaoi and S. Ishio, *Large Magnetoresistance Effect in 82Ni-Fe/Al-Al_2O_3/Co Magnetic Tunneling Junction*, J. Magn. Magn. Mater., **98**, L7 (1991)

[12] T. Yaoi, S. Ishio and T. Miyazaki, *Dependence of Magnetoresistance on Temperature and Applied Voltage in a 82Ni-Fe/Al-Al_2O_3/Co Tunneling Junction*, J. Magn. Magn. Mater., **126**, 430 (1993)

[13] Y. Suezawa and Y. Gondo, *Spin-Polarized Electrons and Magnetoresistance in Ferromagnetic Tunnel Junctions and Multilayers*, J. Magn. Magn. Mater., **126**, 524 (1993)

[14] T. Miyazaki and N. Tezuka, *Giant Magnetic Tunneling Effect in Fe/Al_2O_3/Fe Junction*, J. Magn. Magn. Mater., **139**, L231 (1995)

[15] J. S. Moodera, L.R. Kinder, T.M. Wong, and R. Meservey, Phys. Rev. Lett., **74**, 3273 (1995)

[16] W. J. Gallagher et. al., *Microstructured Magnetic Tunnel Junctions*, J. Appl. Phys., **81**, 3741 (1997)

[17] S. A. Rishton et. al., *Magnetic Tunnel Junctions Fabricated At Tenth-Micron Dimensions by Electron Bean Lithography,* Microelectronic Engineering, **35**, 249 (1997)

[18] Yu Lu et. al., *Bias Voltage and Temperature Dependence of Magnetotunneling Effect*, J. Appl. Phys., **83**, 6515 (1998)

[19] R. H. Koch et. al., *Magnetization Reversal in Micron-Sized Magnetic Thin Films*, Phys. Rev. Lett., **81**, 4512 (1998)

[20] Yu Lu, *An Experimental Study of Ferromagnet-Insulator-Ferromagnet Tunnel Junctions*, Brown University, 1997

Memory Application Considerations

[21] D. D. Tang et. al., *Spin-Valve RAM Cell*, IEEE Trans. Magn., **31**, 3206 (1995)

[22] Z. G. Wang and Y. Nakamura, *Design, Simulation, and Realization of Solid State Memory Element Using the Weakly Coupled GMR Effect*, IEEE Trans. Magn., **32**, 520 (1996)

[23] J. M. Daughton, *Magnetic Tunneling Applied to Memory*, J. Appl. Phys., **81**, 3758 (1997)

Chapter 2

Computer Simulations

2.1 Magnetization Dynamics of a Discrete Lattice

In our computer simulation studies, a thin film of ferromagnetic metal is represented as a discrete lattice of interacting dipoles. The dynamics of such a system is readily described by the Landau-Lifshitz-Gilbert (LLG) equation, which we will consider first. We shall then focus on the interactions that contribute to the total effective field as felt by each dipole in the lattice. A discussion of the specific details of our particular implementation of the LLG equation to simulate magnetization dynamics follows next. We will conclude with a presentation of some simulation results and their implications on memory design with MTJs.

2.1.1 The LLG equation

The Landau-Lifshitz-Gilbert equation is a mechanical law that relates the time rate of angular momentum change to the applied torque. For a dipole with moment $\mathbf{m}$ and effective gyromagnetic ratio γ in an effective magnetic field $\mathbf{H}^{eff}$, the angular momentum is $\mathbf{m}/\gamma$, and the torque is $\mathbf{m} \times \mathbf{H}^{eff}$. Hence:

$$\dot{\mathbf{m}} = -\gamma \mathbf{m} \times \mathbf{H}^{eff} \tag{2.1}$$

In order for our physical model to be accurate, we need to include dissipative effects as well.

In the simplest and most practical picture, this is accomplished by including a phenomenogical term in $\mathbf{H}^{eff}$ that is proportional to $\dot{\mathbf{m}}/|\mathbf{m}|$. Hence, the modified LLG equation becomes:

$$\begin{aligned} \dot{\mathbf{m}} &= -\gamma \mathbf{m} \times \left(\mathbf{H}^{tot} - \frac{\alpha \dot{\mathbf{m}}}{\gamma |\mathbf{m}|} \right) \\ &= \gamma \mathbf{H}^{tot} \times \mathbf{m} + \alpha \frac{\mathbf{m}}{|\mathbf{m}|} \times \dot{\mathbf{m}} \end{aligned} \tag{2.2}$$

Here, α is a dimensionless constant that determines the strength of the dissipative effects, and $\mathbf{H}^{tot}$ is the total physical magnetic field. Notice that for a small value of α, the time rate of change of the magnetic moment is mainly determined by the $\gamma \mathbf{H}^{tot} \times \mathbf{m}$ term in Equation 2.2 above, which is perpendicular to the plane of the moment and the total field. The $\alpha \dot{\mathbf{m}} \times \frac{\mathbf{m}}{|\mathbf{m}|}$ term, on the other hand, pulls the moment back into the plane of the $\mathbf{m}$ and $\mathbf{H}$. Figure 2-1 below illustrates this effect. Without the dissipative term, the magnetic moment precesses around a circle, with it, the moment's trajectory eventually decays down, aligning the moment with the direction of the total magnetic field.

In order to utilize a computer simulation to integrate out the trajectory of the magnetic moment $\mathbf{m}$ as a function of time, we need to express Equation 2.2 in an explicit form. Hence, one substitutes Equation 2.2 back into itself to get

$$\begin{aligned} \dot{\mathbf{m}} &= \gamma \mathbf{H}^{tot} \times \mathbf{m} + \alpha \frac{\mathbf{m}}{|\mathbf{m}|} \times \left(\gamma \mathbf{H}^{tot} \times \mathbf{m} + \alpha \frac{\mathbf{m}}{|\mathbf{m}|} \times \dot{\mathbf{m}} \right) \\ &= \gamma \mathbf{H}^{tot} \times \mathbf{m} + \alpha \frac{\mathbf{m}}{|\mathbf{m}|} \times \gamma \mathbf{H}^{tot} \times \mathbf{m} + \alpha \frac{\mathbf{m}}{|\mathbf{m}|} \times \alpha \frac{\mathbf{m}}{|\mathbf{m}|} \times \dot{\mathbf{m}} \end{aligned}$$

dividing both sides by $|\mathbf{m}|$, we get

$$\frac{d\hat{\mathbf{m}}}{dt} = \gamma \mathbf{H}^{tot} \times \hat{\mathbf{m}} + \alpha \hat{\mathbf{m}} \times \gamma \mathbf{H}^{tot} \times \hat{\mathbf{m}} + \alpha^2 \hat{\mathbf{m}} \times \hat{\mathbf{m}} \times \frac{d\hat{\mathbf{m}}}{dt} \tag{2.3}$$

Now, $\hat{\mathbf{m}} \cdot \hat{\mathbf{m}} = 1$, so

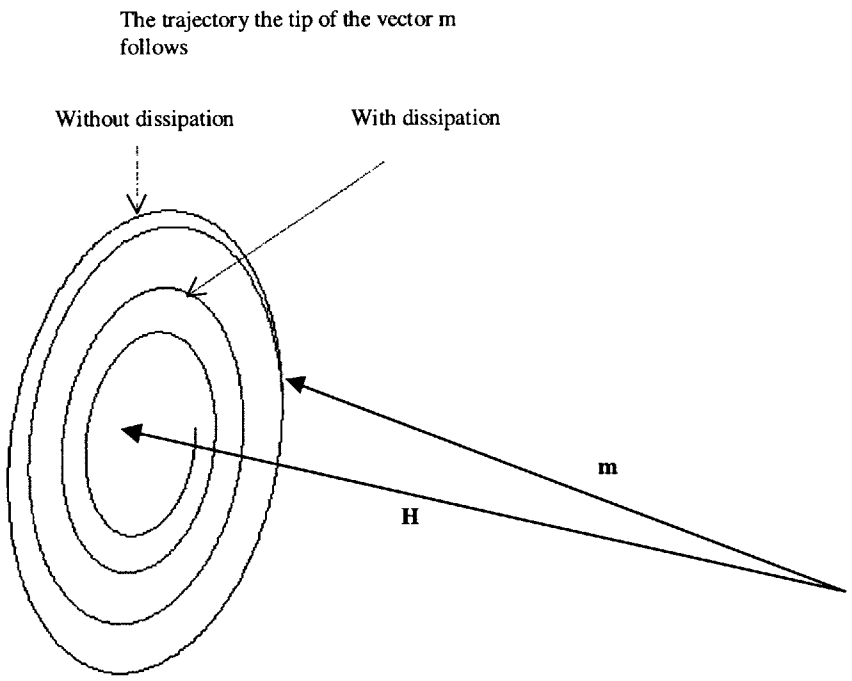

Figure 2-1: The depiction of two trajectories of a single magnetic moment with and without dissipation. The degree of dissipation determines the time it takes for the trajectory to decay away.

$$\begin{aligned} \hat{\mathbf{m}} \times \left(\hat{\mathbf{m}} \times \frac{d\hat{\mathbf{m}}}{dt} \right) &= \left(\hat{\mathbf{m}} \cdot \frac{d\hat{\mathbf{m}}}{dt} \right) \hat{\mathbf{m}} - (\hat{\mathbf{m}} \cdot \hat{\mathbf{m}}) \frac{d\hat{\mathbf{m}}}{dt} \\ &= \frac{1}{2} \frac{d(\hat{\mathbf{m}} \cdot \hat{\mathbf{m}})}{dt} \hat{\mathbf{m}} - \frac{d\hat{\mathbf{m}}}{dt} \\ &= -\frac{d\hat{\mathbf{m}}}{dt} \end{aligned} \tag{2.4}$$

Substituting 2.4 into 2.3 above, and rearranging, we get

$$\frac{d\hat{\mathbf{m}}}{dt} = \frac{\gamma}{1+\alpha^2} \left(\mathbf{H}^{tot} \times \hat{\mathbf{m}} + \alpha \hat{\mathbf{m}} \times \left(\mathbf{H}^{tot} \times \hat{\mathbf{m}} \right) \right) \tag{2.5}$$

Finally, we need to express the LLG equation in terms of the spatial components of the magnetic moment vector.

$$\frac{d\hat{\mathbf{m}}}{dt} = \frac{\gamma}{1+\alpha^2}\left(\begin{vmatrix} \mathbf{i}_x & \mathbf{i}_y & \mathbf{i}_z \\ H_x^{tot} & H_y^{tot} & H_z^{tot} \\ \hat{m}_x & \hat{m}_y & \hat{m}_z \end{vmatrix} + \alpha\hat{\mathbf{m}}\times\begin{vmatrix} \mathbf{i}_x & \mathbf{i}_y & \mathbf{i}_z \\ H_x^{tot} & H_y^{tot} & H_z^{tot} \\ \hat{m}_x & \hat{m}_y & \hat{m}_z \end{vmatrix}\right)$$

Now, evaluating the determinant, one gets:

$$\begin{aligned} \mathbf{H}^{tot}\times\hat{\mathbf{m}} &= \left(H_y^{tot}\hat{m}_z - \hat{m}_y H_z^{tot}\right)\mathbf{i}_x + \left(H_z^{tot}\hat{m}_x - \hat{m}_z H_x^{tot}\right)\mathbf{i}_y + \left(H_x^{tot}\hat{m}_y - \hat{m}_x H_y^{tot}\right)\mathbf{i}_z \\ \hat{\mathbf{m}}\times\mathbf{H}^{tot}\times\hat{\mathbf{m}} &= \begin{vmatrix} \mathbf{i}_x & \mathbf{i}_y & \mathbf{i}_z \\ \hat{m}_x & \hat{m}_y & \hat{m}_z \\ \left(H_y^{tot}\hat{m}_z - \hat{m}_y H_z^{tot}\right) & \left(H_z^{tot}\hat{m}_x - \hat{m}_z H_x^{tot}\right) & \left(H_x^{tot}\hat{m}_y - \hat{m}_x H_y^{tot}\right) \end{vmatrix} \end{aligned}$$

which, when substituted back into equation 2.5 above, yield the equations for the time-evolution of each of the components of the magnetic moment vector:

$$\begin{aligned} \frac{d\hat{m}_x}{dt} &= \frac{\gamma}{1+\alpha^2}\left(\left(H_y^{tot}\hat{m}_z - \hat{m}_y H_z^{tot}\right) + \alpha\left(\hat{m}_y\left(H_x^{tot}\hat{m}_y - \hat{m}_x H_y^{tot}\right) - \hat{m}_z\left(H_z^{tot}\hat{m}_x - \hat{m}_z H_x^{tot}\right)\right)\right) \\ \frac{d\hat{m}_y}{dt} &= \frac{\gamma}{1+\alpha^2}\left(\left(H_z^{tot}\hat{m}_z - \hat{m}_z H_z^{tot}\right) + \alpha\left(\hat{m}_z\left(H_y^{tot}\hat{m}_z - \hat{m}_y H_z^{tot}\right) - \hat{m}_x\left(H_x^{tot}\hat{m}_y - \hat{m}_x H_y^{tot}\right)\right)\right) \\ \frac{d\hat{m}_z}{dt} &= \frac{\gamma}{1+\alpha^2}\left(\left(H_x^{tot}\hat{m}_y - \hat{m}_x H_y^{tot}\right) + \alpha\left(\hat{m}_x\left(H_z^{tot}\hat{m}_x - \hat{m}_z H_x^{tot}\right) - \hat{m}_y\left(H_y^{tot}\hat{m}_z - \hat{m}_y H_z^{tot}\right)\right)\right) \end{aligned} \tag{2.6}$$

The equation 2.6 above describes the time-evolution of a magnetic dipole inside a magnetic field in a dissipative medium. The details of discretizing and integrating this equation are discussed in Appendix A.

Notice that the same set of equations can be used to describe an arbitrary lattice of magnetic dipoles. As long as the spatial discretization is fine enough, a collection of magnetic moments in three dimensional space is sufficient to represent the magnetic behavior of any ferromagnetic material. The only modification to equation 2.6 will then be the introduction of spacial indices – all the components of the magnetic moment and the total magnetic field will be a function

of both space and time.

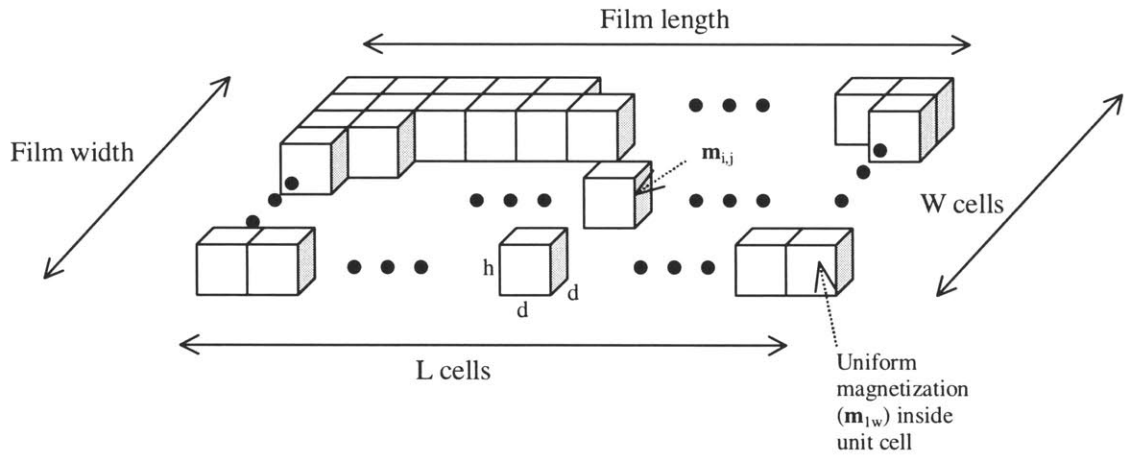

Figure 2-2: The computational lattice.

In the case of such a magnetic lattice, $\mathbf{H}^{tot}$ includes, in addition to the external magnetic field, interaction terms within neighboring moments. In the following section, we investigate these interactions.

2.1.2 Total magnetic field inside a ferromagnet

There are four main contributions to the effective magnetic field inside a ferromagnetic metal. These originate from: the external field, the exchange interactions, anisotropy, and demagnetization effects. We will introduce and discuss each of these contributions briefly below.

External Field

This is simply the applied field. We will denote it as $\mathbf{H}^{ext}$.

Exchange Field

The exchange interaction is a quantum mechanical effect that arises from the symmetry of wavefunctions of a system under the interchange of its particles. This interaction depends only on the total spin of the system, and not on its direction[1]. A general discussion of the

different models of exchange interaction in magnetic materials is beyond the scope of this work[1]. When considering a discrete magnetic lattice as in our simulation, a good approximation to the exchange energy density between two neighboring moments is given by ([2])

$$\begin{aligned} W_{xch} &= \frac{2A_x}{d^2}\left(1 - \hat{\mathbf{m}} \cdot \hat{\mathbf{m}}_1\right) \\ &= \frac{2A_x}{d^2}\left(1 - \frac{m_x m_{1x} + m_y m_{1y} + m_z m_{1z}}{|\mathbf{m}||\mathbf{m}_1|}\right) \end{aligned} \tag{2.7}$$

where $\hat{\mathbf{m}}$ and $\hat{\mathbf{m}}_1$ are the neighboring normalized moments, d is the distance between them, and A_x is the macroscopic exchange stiffness coefficient ([?],[5]). Both our experimental and simulation work has focused primarily on MTJ devices having electrodes with uniform magnetic properties[2]. For the rest of the discussion, it is assumed that the computational lattice is taken to be a uniform magnetic material; hence, $|\mathbf{m}| = |\mathbf{m}_1| = M$.

Now, using $W_{xch} = -\mathbf{H}^{xch} \cdot \mathbf{m} = -\left(H_x^{xch} m_x + H_y^{xch} m_y + H_z^{xch} m_z\right)$, we have

$$\begin{aligned} H_j^{xch} &= -\frac{\partial W_{xch}}{\partial m_j} \\ &= \frac{2A_j}{d^2}\left(\frac{m_{1j}}{M^2}\right), \text{ with } j = x, y, z \end{aligned} \tag{2.8}$$

This is the field acting on the moment at the origin due to a neighboring moment a distance d away. For a computational lattice, the exchange field acting on each moment due to its neighbors is summed up vectorally to yield the net field due to exchange interactions. Since the lattice in the micromagnetics simulator is a single layer in the z direction and the unit cells have the same dimensions in the x and y axes (see Figure 2-2), it follows that $A_x = A_y = A$; $A_z = 0$. The effective magnetic field acting on a single moment due to exchange interactions with its neighbors is then given by

[1] See [3] for background on models of exchange interaction on different magnetic materials.

[2] Magnetic impurities inevitably exist in a thin film – for example, due to variation in the magnetic properties of the material resulting from local stresses, or thanks to minor contamination of the electrode material during processing. In a careful fabrication process, however, these effects can be mitigated, and the end effect of the impurities on the electrode's magnetic properties can be minimized.

$$H_j^{xch} = \sum_l \frac{2A}{l^2}\left(\frac{m_{1j}}{M^2}\right)$$

Physically, exchange interactions act on the atomic scale, and they are short ranged. The dependence on the inverse square of the distance also justifies mathematically an approximation by truncating the interacting neighbors for each magnetic moment. Generally, a next-nearest neighbor approximation is more than enough to capture the basic exchange interactions in a magnetic lattice[3], as shown below.

$$H_j^{xch} = \sum_{n.n.neighbors} \frac{2A}{l^2}\left(\frac{m_{1j}}{M^2}\right)$$

Anisotropy Field

Anisotropy energy arises from the relativistic interactions of the electons – namely, the spin-orbit, and spin-spin interactions ([1], [6]). These are of the order $\frac{v^2}{c^2}$, hence, their effects are usually small, and are calculated using quantum perturbation theory, applied to the Hamiltonian of the ferromagnetic crystal. Since the angular momenta of electrons inside a ferromagnetic material are correlated to the bond directions and the crystal structure, anisotropy energy shows a dependence on the symmetry axes of the ferromagnetic crystal. In other words, it is a function of the direction of the total magnetic moment inside the material. Of course, there are other reasons for magnetic anisotropy inside a thin film, such as magnetostriction ([3]), or it can just be induced during processing by the application of a strong external magnetic field.

For a ferromagnetic thin film with uniaxial anisotropy, the anisotropy energy density associated with a single moment $\mathbf{m}$ can be approximated by ([2])

$$W_{ans} = K_u\left[1 - (\hat{\mathbf{m}} \cdot \hat{\mathbf{m}}_0)^2\right] \tag{2.9}$$

[3]Such an approximation is also justified computationally: a general sum of exchange interactions over the entire lattice would be $O\left(N^2\right)$, whereas a next-nearest neighbor approximation would basically be $O\left(N\right)$.

where $\hat{\mathbf{m}}_0$ is the unit vector along the direction of the "easy axis" of the film. Notice that $\hat{\mathbf{m}}_0$ gives the "easy axis" direction because the anisotropy energy is a minimum when the moment $\mathbf{m}$ is parallel (or antiparallel) to $\hat{\mathbf{m}}_0$; hence, the effective anisotropy field tends to align the moment along this direction.

For simplicity, we shall choose our coordinate system so that the x-y plane gives the plane of the thin film, and the x-axis is $\hat{\mathbf{m}}_0$. Using the approach as in our discussion with the exchange energy, we have

$$\begin{aligned} H_j^{ani} &= -\frac{\partial W_{ani}}{\partial m_j} \\ &= -\frac{\partial}{\partial m_j}\left(K_u\left[1-\left(\frac{m_x}{M}\right)^2\right]\right) \\ &= \begin{cases} \frac{2K_u m_x}{M^2} & , \text{if } j = x \\ 0 & , \text{otherwise} \end{cases} \end{aligned}$$

Demagnetization Effects

The long range dipole-dipole interactions between magnetic moments in a ferromagnet give rise to an effective demagnetization field. The demagnetization field on a single moment $\mathbf{m}$ due to another point-like moment $\mathbf{m}_1$ is given by ([7])

$$\mathbf{H}^{dmag} = \frac{3\mathbf{n}\left(\mathbf{n}\cdot\mathbf{m}_1\right) - \mathbf{m}_1}{r^3} \tag{2.10}$$

where $\mathbf{n}$ is the unit vector pointing from $\mathbf{m}$ to $\mathbf{m}_1$.

The brute force way to implement the demagnetizing effects in a computer simulation would involve calculating the effect of each discrete moment block inside the computational lattice on each of the other moments. Notice that such a calculation would be $O\left(N^2\right)$, where N is the number of unit cells used to represent the ferromagnet in the simulation. Also, since dipole-dipole interactions are long-ranged, a realistic representation of a ferromagnet has to include the demagnetization field due to all the discrete moment blocks. Hence, calculation of the demagnetization field is a computational bottleneck for any realistic micromagnetics simulator.

It turns out that this bottleneck can be partially mitigated if one uses fast fourier transforms (FFTs) to calculate the demagnetization field everywhere inside the lattice. Consider Figure 2-3 below.

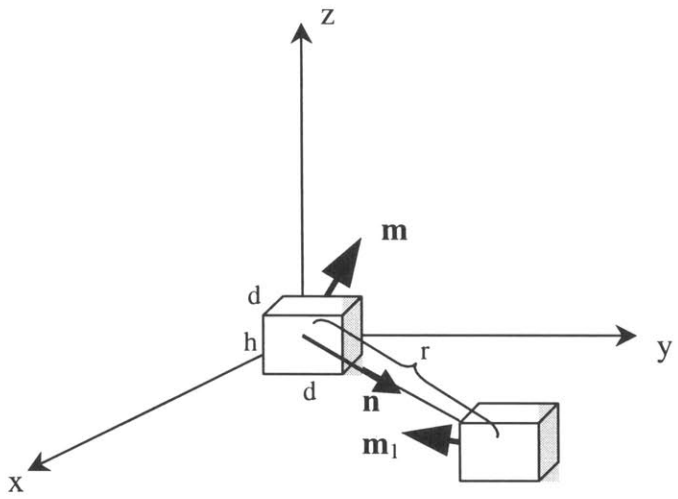

Figure 2-3: Two unit cells of a micromagnetic lattice. The magnetization inside each cell is uniform, given by $\mathbf{m}$ and $\mathbf{m}_1$ respectively.

Since each of the two unit cells shown above have a uniform magnetization, we need to express equation 2.10 in the continuous limit. The effective demagnetization field (along the i direction) on the cell centered at $\mathbf{x}$ due to the magnetization at $\mathbf{x}_1$ (along the j direction) is given by

$$H_{ij}^{dmag}(\mathbf{x}) = \int_{unitcell} d^3\mathbf{x}_1 G'_{ij}(\mathbf{x}-\mathbf{x}_1)\, m_j(\mathbf{x}_1) \tag{2.11}$$

where $G'_{ij}(\mathbf{r})$ is the dipole-dipole interaction kernel, given by

$$G'_{ij}(\mathbf{r}) = \frac{3\hat{r}_i\hat{r}_j - \delta_{ij}}{r^3} \quad ; \quad \mathbf{r} = \mathbf{x}-\mathbf{x}_1 \quad \text{and } i,j = x,y,z$$

Now, for simplicity, let us define the kernel coefficients $G_{ij}(\mathbf{x})$ as follows:

$$G_{ij}(\mathbf{x}) = \int_{-h/2}^{h/2} dz_1 \int_{-d/2}^{d/2} dy_1 \int_{-d/2}^{d/2} dx_1 G'_{ij}(\mathbf{x} - \mathbf{x}_1)$$

Notice that by the setup of the problem, magnetization vector is uniform inside a unit cell; hence it comes out of the integral in equation 2.11. The total demagnetization field at location $\mathbf{x}$ is then given by summing up all the contributions in each direction, due to all unit cells.

$$\begin{aligned}
H_y^d(\mathbf{x}) &= M \int_{lattice} d^3\mathbf{x}_1 \left[G_{xx}(\mathbf{x} - \mathbf{x}_1)\hat{m}_x(\mathbf{x}_1) + G_{xy}(\mathbf{x} - \mathbf{x}_1)\hat{m}_y(\mathbf{x}_1) + G_{xz}(\mathbf{x} - \mathbf{x}_1)\hat{m}_z(\mathbf{x}_1)\right] \\
H_y^d(\mathbf{x}) &= M \int_{lattice} d^3\mathbf{x}_1 \left[G_{yx}(\mathbf{x} - \mathbf{x}_1)\hat{m}_x(\mathbf{x}_1) + G_{yy}(\mathbf{x} - \mathbf{x}_1)\hat{m}_y(\mathbf{x}_1) + G_{yz}(\mathbf{x} - \mathbf{x}_1)\hat{m}_z(\mathbf{x}_1)\right] \\
H_z^d(\mathbf{x}) &= M \int_{lattice} d^3\mathbf{x}_1 \left[G_{zx}(\mathbf{x} - \mathbf{x}_1)\hat{m}_x(\mathbf{x}_1) + G_{zy}(\mathbf{x} - \mathbf{x}_1)\hat{m}_y(\mathbf{x}_1) + G_{zz}(\mathbf{x} - \mathbf{x}_1)\hat{m}_z(\mathbf{x}_1)\right]
\end{aligned} \tag{2.12}$$

Since dipole-dipole interaction is a function of distance only, there are only six independent kernels in the expressions above: G_{xx}, $G_{xy}(= G_{yx})$, $G_{xz}(= G_{zx})$, G_{yy}, $G_{yz}(= G_{zy})$, G_{zz}. The equations in 2.12 are linear combinations of convolution integrals. Hence, their Fourier transforms become a linear combination of products of transforms:

$$\begin{aligned}
\tilde{H}_x^d(\mathbf{k}) &= M\left(\tilde{G}_{xx}(\mathbf{k}) * \tilde{m}_x(\mathbf{k}) + \tilde{G}_{xy}(\mathbf{k}) * \tilde{m}_y(\mathbf{k}) + \tilde{G}_{xz}(\mathbf{k}) * \tilde{m}_z(\mathbf{k})\right) \\
\tilde{H}_y^d(\mathbf{k}) &= M\left(\tilde{G}_{yx}(\mathbf{k}) * \tilde{m}_x(\mathbf{k}) + \tilde{G}_{yy}(\mathbf{k}) * \tilde{m}_y(\mathbf{k}) + \tilde{G}_{yz}(\mathbf{k}) * \tilde{m}_z(\mathbf{k})\right) \\
\tilde{H}_z^d(\mathbf{k}) &= M\left(\tilde{G}_{zx}(\mathbf{k}) * \tilde{m}_x(\mathbf{k}) + \tilde{G}_{zy}(\mathbf{k}) * \tilde{m}_y(\mathbf{k}) + \tilde{G}_{zz}(\mathbf{k}) * \tilde{m}_z(\mathbf{k})\right)
\end{aligned} \tag{2.13}$$

The coefficients G_{ij} depend only on the geometry of the problem, and do not evolve in time. This means that they can be calculated beforehand, and their Fourier transforms can be stored for subsequent computations during the integration of the LLG equation[4]. Also,

[4]See Appendix B for a discussion about the calculation of these geometrical factors.

the multiplication and addition operations in equations B.1 require negligible computation time, compared with the integrations that need to be performed in equations 2.12. Therefore, using FFTs to compute the demagnetization field is a great deal more efficient than using an algorithm based on time-domain calculations only. The main bottleneck in an FFT approach then becomes the time required to compute Fourier transforms and their inverses. It turns out that, with an efficient set of FFT algorithms, the overall calculation of the demagnetization field can be made $O(N \log N)$.

However, efficiency comes at the price of need for extra memory to store the geometrical factors, as well as the Fourier transforms of the magnetization and field matrices. Moreover, since our magnetic lattice is non-periodic and finite, there will be contributions from periodic images. The remedy of this problem requires the use of a computational lattice that is at least twice as big in each dimension than the original block[5]. The larger computational lattice contains the original magnetization values in one quadrant, and zero elsewhere. The inverse transforms of the matrices $\tilde{H}_i^{dmag}(\mathbf{k})$ are then truncated to yield the demagnetization field matrices $H_i^{dmag}(\mathbf{x})$.

Total Effective Magnetic Field

The total field inside a ferromagnet is given by the sum of magnetic fields due to anisotropy, exchange, and demagnetization, as well as the externally applied field.

$$\mathbf{H}^{tot} = \mathbf{H}^{ext} + \mathbf{H}^{ani} + \mathbf{H}^{xch} + \mathbf{H}^{dmag} \tag{2.14}$$

Inserting equation 2.14 into equation 2.6 above, we get the full Landau-Libshiftz-Gilbert formulation for the evolution of magnetization inside any ferromagnetic material.

[5] See [3] for details on periodic images on FFTs.

2.2 The LLG Simulator

The computer program to simulate the magnetization dynamics of a ferromagnetic thin film electrode has been based on the equations and algorithms for integrating the Landau-Lifshitz-Gilbert equation as presented above. In this section, we will introduce the program and the details of our particular implementation.

2.2.1 Graphical User Interface

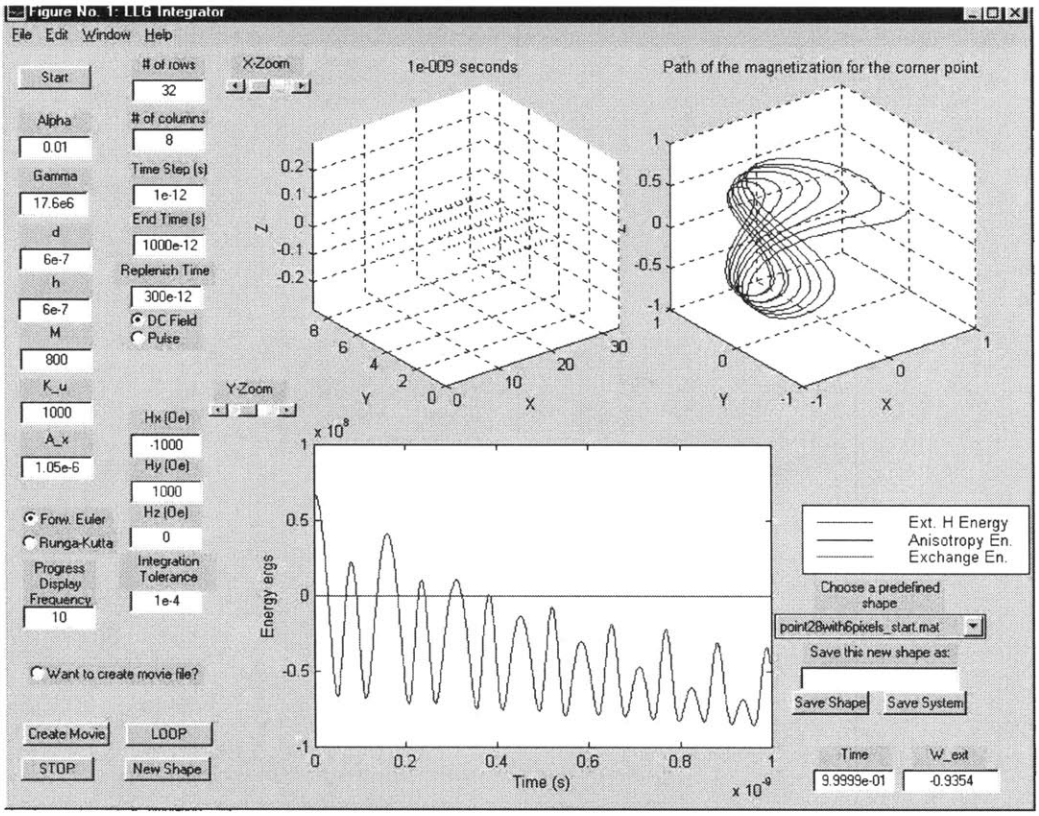

Figure 2-4: The main graphical user interface window of the LLG simulator.

The front-end of the computer program is designed using MATLAB^{TM} graphical user interface (GUI) controls. MATLAB^{TM} has been chosen to be the main interface platform, not only because its GUI's are very simple and fast to create, but also because it allows the simulation data to be analyzed quickly and efficiently. Despite the convenience of its high-level script

language, MATLAB^{TM} is not fast or effective enough for large-scale computations, such as the ones required by an LLG simulator. Hence, the computation is performed by a C program that is compiled to share memory space with MATLAB^{TM}. This way, the simulation program is made highly reconfigurable without comprimising speed and efficiency.

The main GUI window (shown above) serves to input the variables that will be used during the actual dynamics calculations; it also displays some of the relevant results of the simulation once the calculations are completed.

The first seven input fields on the left determine the magnetic properties of the thin film – these are the damping coefficient (α), the gyromagnetic ratio (γ), the length and the width of the unit cells (d), the thickness of the ferromagnetic film (h), magnetic moment strength (M), uniaxial anisotropy coefficient (K_u), and exchange coefficient (A). The size of the lattice, the integration time length, and the strength and the direction of the external magnetic field, as well as its duration, are also controlled through this window. The top left graph displays the final distribution of magnetic moments, whereas the top right plot shows the time evolution of average magnetization of the film in three dimensions. The bottom graph exhibits the external, anisotropy and exchange energy densities as a function of time. The GUI allows the loading and saving of different lattice shapes, as well as magnetic moment distributions.

New computational lattice shapes are defined interactively in a pop-up window, by selecting or deselecting unit cells through the click of a mouse.

The hexagonal shape shown in Figure 2-5 has been the principal form of interest in this study. This is mainly because all the experimental data comes from similar hexagonal forms. Since one of the goals of the simulation is to attempt to explain features observed in the experimental data, this choice of shape is justified. For the rest of this work, it will be assumed that any computational or experimental data that we present comes from the study of such a hexagonal shape, unless specified otherwise.

For the purposes of comparing simulation results directly with the experiment, another pop-up window has been created to study magnetic hysteresis loops of the thin films (see Figure 2-6 below). This window contains the control inputs for the range of fields to be applied. By controlling the range and duration of the easy-axis and hard-axis fields, the effects of all the different setups in the experiments performed can be simulated.

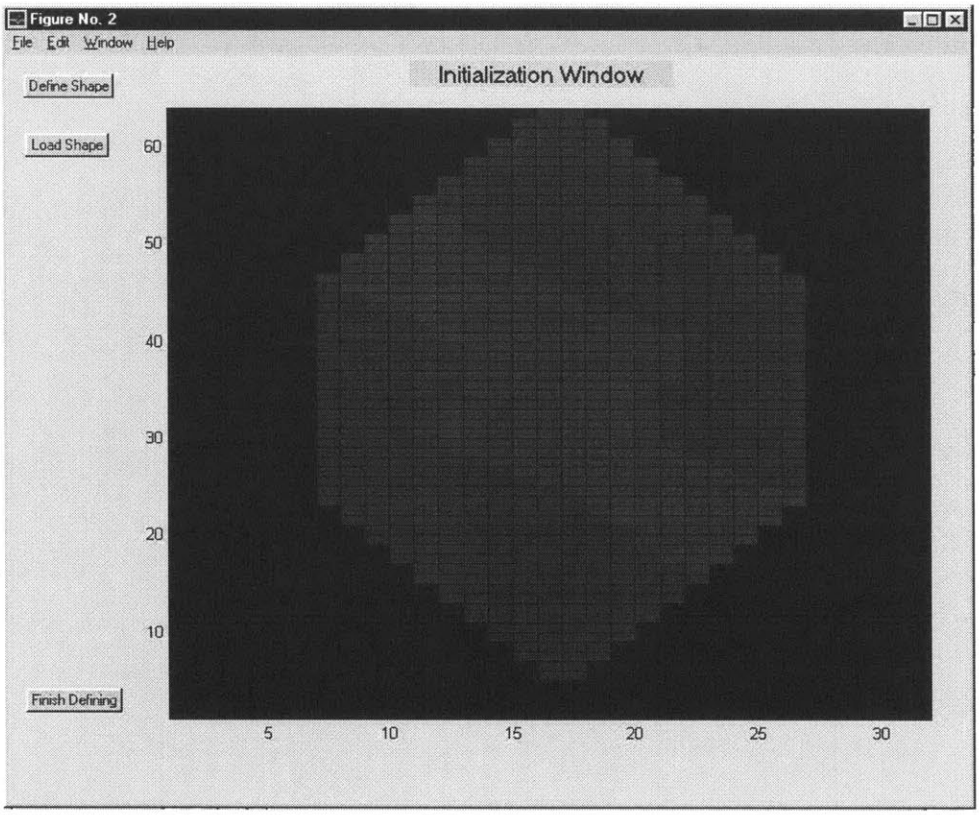

Figure 2-5: The window for defining new lattice shapes. The axes denote the indices to the unit cells. The easy axis is parallel to the length (top to bottom) of the junction.

The last pop-up window is used to create movie files representing the time evolution of the magnetization distribution inside the lattice. Movies are created in MATLAB^{TM} movie format. They can be saved either as MATLAB^{TM} or MPEG movies.

2.2.2 The Integrator

As mentioned above, the actual computations involved in integrating the LLG equation are performed in a C program that is compiled as a dynamic link library (*dll*) file. This program is directly callable within MATLAB^{TM} as a native function; hence, it shares the same memory space and its results are returned in the MATLAB^{TM} workspace. The source code of the simulator is given in Appendix C, together with the MATLAB^{TM} codes associated with the graphical user interface.

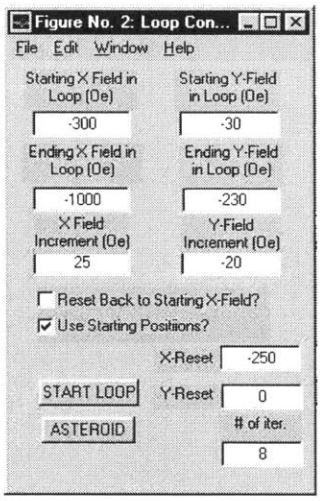

Figure 2-6: Loop window. This part of the GUI allows the user to simulate magnetic hysteresis loops.

Below is a simplified block diagram that describes the LLG simulation program.

The "replenish" time as shown in Figure 2-8 above is a detail associated with the particular integration technique that has been used in the simulator; it is discussed in Appendix A among other issues related to integrating the LLG equation.

2.3 Simulation Results

In this section, we will present a set of systematic simulation studies that pertain to the question of magnetization reversal in ferromagnetic thin films. In particular, we will attempt to characterize the dependence of the switching thresholds on the applied field direction, and its duration. Both of these properties are of immediate significance for MRAM applications – not only for learning more about the physics of magnetization reversal so that better MTJs could be produced, but also for determining the design constraints involved so that effective engineering trade-offs could be made.

The simulation results presented here concentrate on a permalloy-like material, with $M =$ 800 emu/cm^3, $K_u = 1000$ erg/cm^3, $A = 1.05x10^{-6}$ erg/cm, and $\gamma = 17.6$ MHz.

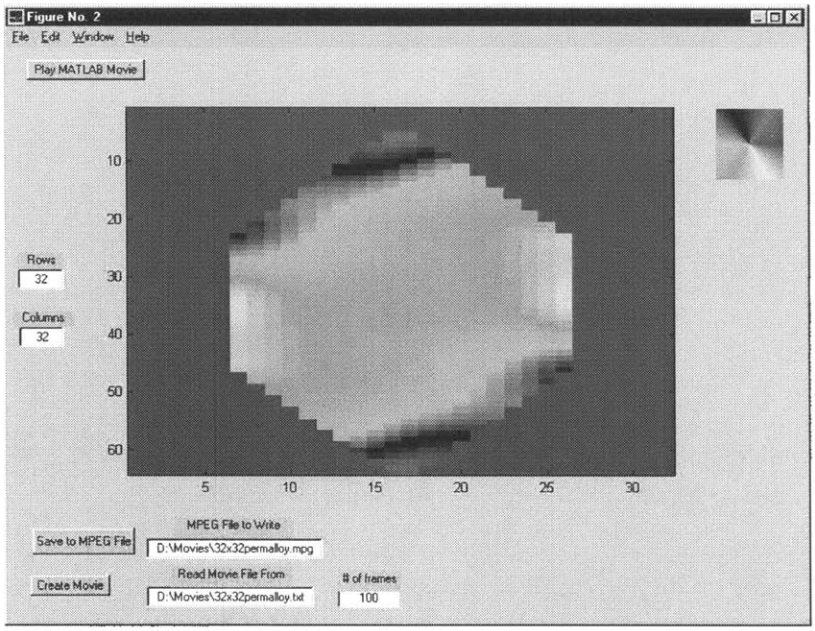

Figure 2-7: Evolution of the magnetization distribution inside the ferromagnetic thin film can be displayed in this movie window.

2.3.1 Asteroid Simulations

The dependence of switching characteristics on the applied field direction will determine the overall configuration geometry of an MRAM array composed of magnetic tunneling junctions. The desired directional dependence is summarized by the so-called Stoner-Wohlfarth asteroid curve ([9]). Such a symmetric asteroid curve allows an orthogonal configuration of the word and bit lines to select a specific bit in a MRAM device, maximizing disturbance rejection and areal density. However, the Stoner-Wohlfarth asteroid presents too simplistic a picture to be of any practical use; it assumes a tiny, single domain magnetic particle with uniaxial anisotropy, and ignores demagnetization effects altogether. In this computational study, we have attempted to adopt a more realistic picture. Here, we will present certain results pertaining to the effect of size, shape and the damping parameter of a ferromagnetic thin film on its "asteroid" curve.

Each computation starts from an initial magnetization distribution throughout the lattice. This starting state is usually determined by the application of a large field in the positive x-direction (parallel to the easy-axis), and later turning the field off to let the magnetization

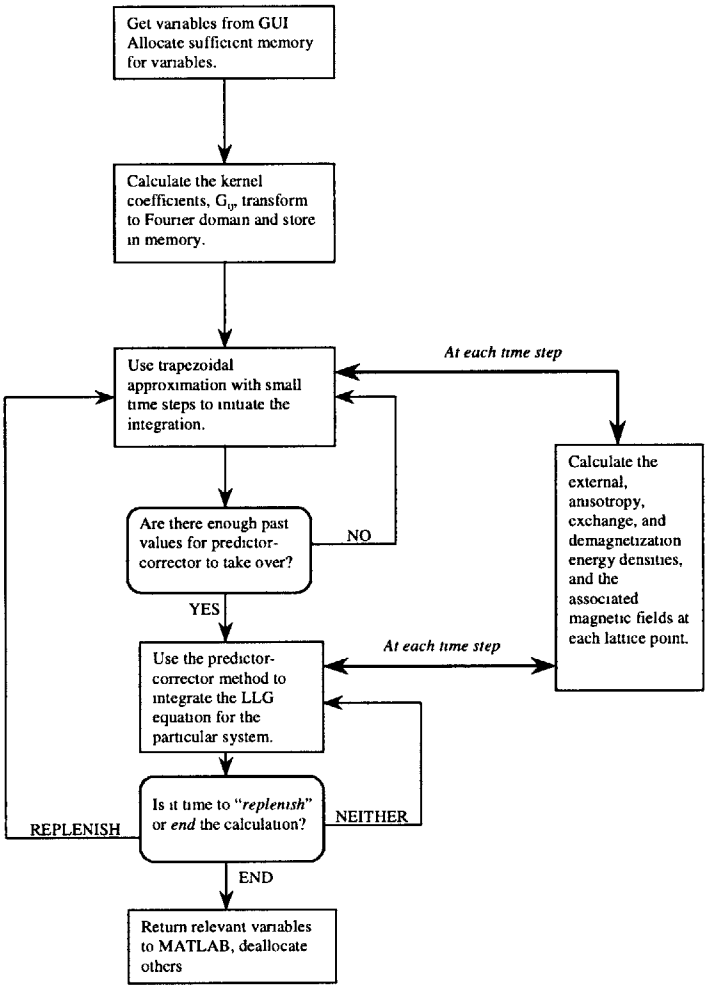

Figure 2-8: The block diagram of the LLG simulation program.

relax into an equilibrium. This process generally yields a total magnetization whose normalized component in the positive x-direction is more than 0.99. The resultant distribution of magnetic moments is then saved and used as the initial magnetic state for subsequent calculations on that particular lattice.

For a given computational lattice, the asteroid curve is determined by localizing the easy-axis switching field at each value of $H_{\perp}$. The localization procedure follows a binary search algorithm that finds the switching threshold by comparing the final magnetization distribution to the starting, "unswitched" state. "Switching" is defined as the final normalized magnetization having a negative x-direction component of more than 0.8. After each computation, the program

automatically adjusts the external field value to zoom in on the coercive field (H_c), until after about 8 iterations, it finds the threshold within better than 3 Oe accuracy. The same procedure is repeated for each hard-axis field value indicated in the control window.

Figure 2-9 below shows the results of one study that is aimed at understanding the effect of the damping parameter on the asteroid curves. The thin film electrode simulated is a 0.036 × 0.108 μm^2 hexagon that is 60 Å thick.

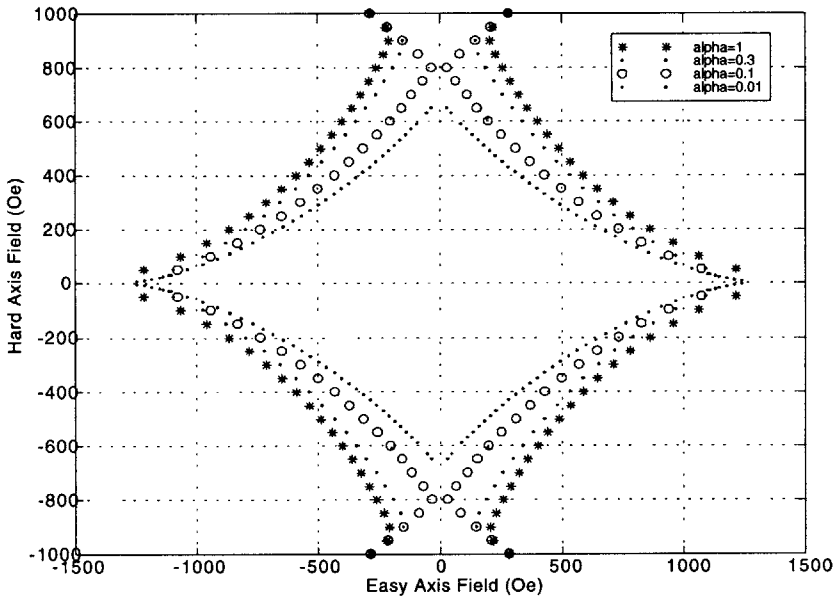

Figure 2-9: Switching thresholds as a function of the easy and the hard-axis fields applied, for various values of the damping parameter α. The junction size is 0.036 × 0.108 μm^2.

It should be noted that for $H_\perp$ values above a certain level, magnetization of the thin film tends to have a strong hard-axis component, and it cannot switch up to 80 % in the negative x-direction. Beyond this point, $H_\parallel$ must increase at the same rate as $H_\perp$ to yield 80 % switching along the easy-axis. That is why the values depicted in Figure 2-9 above a certain hard-axis field seem to stray away from the asteroid shape, following a linear pattern. Notice that our definition is just a matter of convention; redefining the switching threshold as $\bar{m}_x < 0$ restores the continuity in the asteroid shape. However, the algorithm implemented needs to make sure

magnetization reversal takes place with the system reaching an equilibrium, and the 80 % limit on switching provides a convenient guarantee that the system will reach a stable equilibrium with the magnetization switched.

It is evident from Figure 2-9 that there is a systematic reduction in the asteroid curves along $H_\perp$ with decreasing α. This is the first time such a relationship is demonsrated between the damping parameter of a ferromagnetic thin film electrode and its switching characteristics as a function of external field direction. Notice that the asteroid curve that corresponds to the largest value of alpha is also the one that is most symmetric with respect to the interchange of $H_{||}$ and $H_\perp$ axes. In a sense, the material with $\alpha = 1$ is very close to a Stoner-Wohlfarth particle. Our results suggest that, if symmetric directional switching characteristics are required, choosing a free layer electrode with enough magnetic damping is crucial.

Interestingly enough, recent experimental literature indicates that the damping parameter for most of the ferromagnetic alloys that are regarded as candidates for MRAM applications is probably as low as 0.01, if not smaller. In this light, the findings in Figure 2-9 become especially relevant and significant, pointing out a potential problem for the development of MRAM. This finding has not yet been reported in experimental studies, mainly because such works have almost exclusively focused on magnetic hysterisis loops obtained by slowly sweeping an external field. This scenario – that we shall name as the *static case* – allows the total magnetic moment of the film enough time to reach an equilibrium for each incremental change in the applied field. The resulting macroscopic behavior, characterized by slow resistance measurements on MTJs taken long after oscillations of the magnetization vector damps out, is similar to what happens for large α. In the simulations discussed here, however, we have studied a sudden application of a field similar to what would happen in a MRAM configuration where current pulses will be used to write bits on MTJs – a scenario that we name as the *dynamic case.*

The results shown in Figure 2-10 are from a similar simulation, where only the length of the free layer has been reduced. The electrode is now a 0.036×0.072 μm^2 hexagon[6] (aspect ratio 1:2).

Changing the aspect ratio of the electrode does not seem to have a large impact on its

[6]Most of the junctions produced by our group have free layers that are 60 Å thick; so, our simulation efforts have also concentrated on ferromagnetic electrodes with this thickness. Henceforth, it will be assumed that the material under study has this particular thickness.

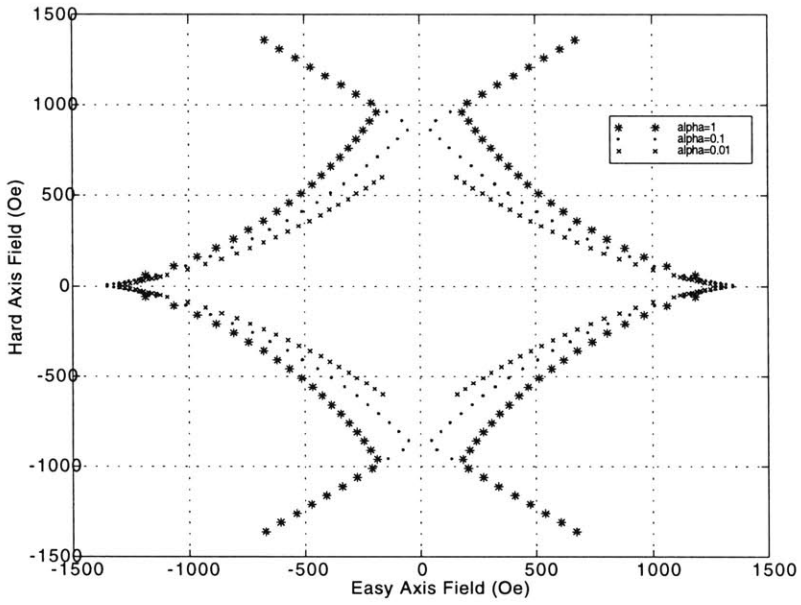

Figure 2-10: Switching thresholds as a function of the easy and hard axis fields applied, for a 0.036×0.072 μm^2 junction.

directional switching characteristics, and the strong dependence on the damping parameter is still evident.

A variation on the total area of the junction, however, creates a more noticable change on the asteroid curves. As the junction area gets smaller, the dominant factor that determines H_c becomes the demagnetization field inside the electrode. Both of the free layers in Figures 2-9 and 2-10 are very small and their magnetization dynamics are dominated by demagnetization effects. That is why there is little difference between their asteroid curves.

On the other hand, Figure 2-11 shows the simulation data from progressively larger free layers, all with the same aspect ratio. As the junction area increases, the boundaries of the electrode separate further apart and the coercive field starts to get governed more by local behavior (i.e., the exchange interaction), as well as the anisotropy field, which is assumed constant throughout the lattice. For all the asteroid curves shown in Figure 2-11, $\alpha = 0.01$.

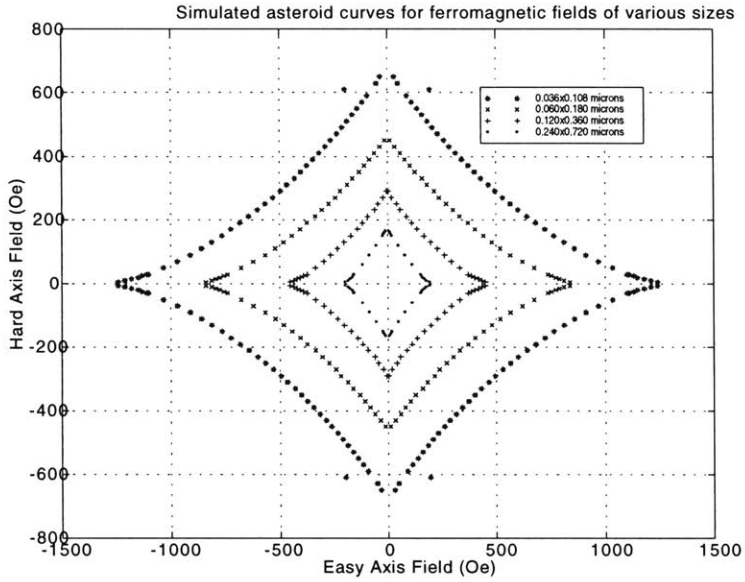

Figure 2-11: Switching thresholds for various field directions and junction sizes. The damping parameter is 0.01 in all cases.

Notice that the coercive fields decrease with increasing junction size. Also the directional switching characteristics tend to become more symmetric for a larger area junction. This finding is consistent with earlier experimental results of our group, in which the same behavior was observed, though for the static case, for various rectangular junctions with aspect ratios of 1:8[7].

The collection of simulation data on directional switching characteristics is not complete to determine the entire range of possible engineering design parameters. They do, however, imply somewhat tight bounds for certain issues. For instance, Figure 2-11 suggests that lower magnetic field pulses (i.e. lower currents) will be necessary to write bits on larger MTJs. Hence, a larger memory element means less power dissipation, and less heating. However, larger devices give rise to a smaller bit density, making MRAM less desirable. Smaller devices also mean that the dynamic asteroid curve of each MTJ will be more non-symmetric, leading to difficulties in

[7]Data courtesy of Dr. Yu Lu. See [10] for a study on magnetic switching at low frequencies.

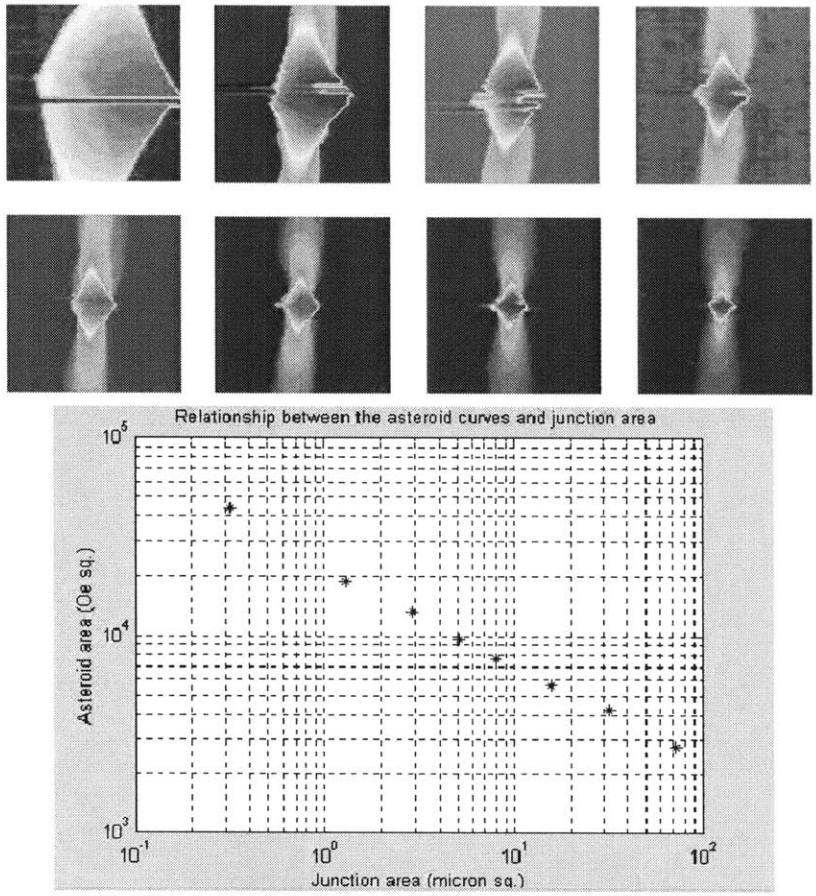

Figure 2-12: The experimentally observed dependence of directional switching characteristics on junction size. In the asteroid plots, brighter colors depict a larger resistance change.

the orthogonal bit selection technique and increasing the junctions' sensitivity to disturbance. Moreover, Figures 2-9 and 2-10 suggest that the damping coefficient of junction free layers is also a determining factor in directional switching characteristics, particularly for smaller junctions. Hence, not only size and geometry, but also the material of the MTJ electrodes will be a major factor in our attempts to achieve the desired Stoner-Wohlfarth asteroid.

2.3.2 Pulse Simulations

In this part of our computational study, we set out to determine the dependence of switching thresholds on the width of an applied magnetic pulse. This inquiry is part of our efforts to

characterize the switching speeds of MTJs. Here, we will present a limited set of results that will be relevant for the discussion of the experimental findings in the next chapter.

Standard semiconductor processing techniques had to be modified, and new technologies had to be invented to produce the magnetic tunneling junctions efficiently and effectively. One of the main challenges still faced today is the difficulty of defining exact shapes for MTJs at sub-micron scales. To estimate how shape variations due to processing might effect the magnetization dynamics of the junctions, we studied magnetization reversal in certain "distorted" shapes and compared our results to those obtained from "nominal" shapes. Figure 2-13 shows a comparison of a "nominal" versus "distorted" shape for a $0.060x0.180$ μm^2 junction. The total number of unit cells in each computational lattice is the same, hence, they both represent the same total junction area. The degree of distortion on the shape is chosen to be consistent with the images of real junctions obtained using atomic force microscopy (AFM).

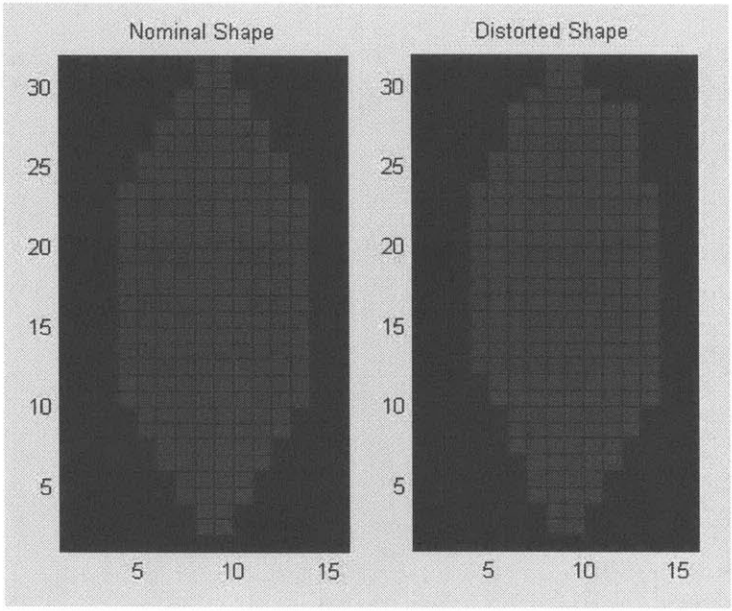

Figure 2-13: A comparison of two computational lattice shapes used in this study, for a 0.060×0.108 μm^2junction.

Numerical pulse width studies are conducted as follows: we start with the lattice magnetized along the positive x-direction, and apply an external magnetic field of a given duration to switch the magnetization in the opposite direction. This procedure is repeated for a set of easy-axis

field values that covers the switching threshold of the electrode. $H_{\perp}$ is set to -100 Oe for simplicity and consistency throughout this study. The magnetic state is integrated out for a time (about 5 nanoseconds or more) that is long enough for any oscillations in the magnetization to die out. This way, one of the switching thresholds can be captured and studied in detail. Figure 2-14 below shows the results of such a study on the "normal" shape described above.

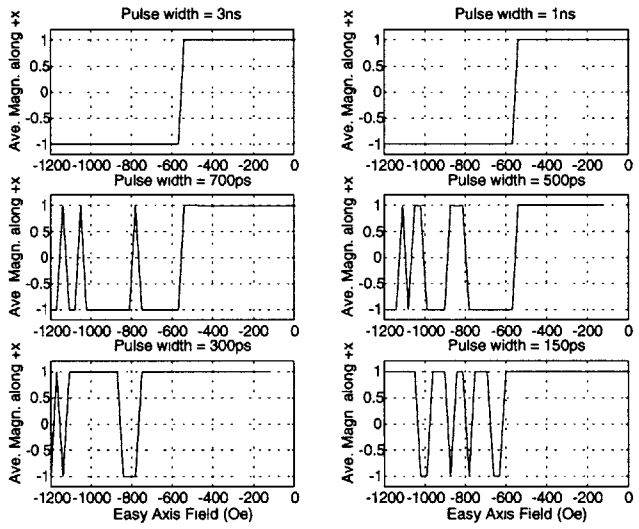

Figure 2-14: Pulsed magnetic field sweeps on the nominal shape for various pulse widths. The damping parameter is taken to be 0.01.

For a long enough magnetic pulse, the pulse sweep yields a clean switching threshold. As the width starts to get comparable to the resonant frequency of the free layer – which is a function of the applied field, and α, among other material dependent factors – interesting things start to happen. Certain values of the easy-axis field above the switching threshold do not reverse the magnetization. This finding is completely associated with the dynamic case; it does not occur with relatively slow external field sweeps using electromagnets. We believe the explanation for this phenomenon lies in the fact that an external field alters the magnetic potential seen by the total magnetic moment of the electrode. A particular field value and pulse width can sometimes leave the magnetic moment in such a place that, with the sudden termination of the external field, the potential changes in a way that favors the return of the moment back to the

unswitched state.

This particular behavior by the total magnetic moment has indeed been observed during the simulations. Figure 2-15 illustrates this phenomenon. Notice that the plot on the right in

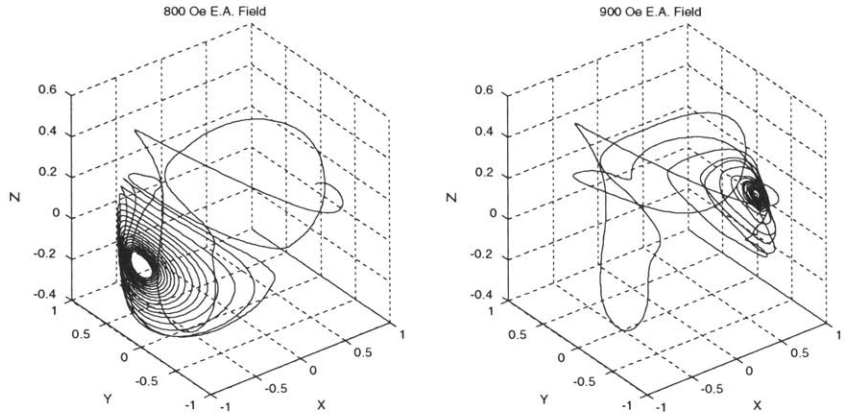

Figure 2-15: The trajectory of the total magnetic moment of the $0.060 \times 0.180\ \mu\text{m}^2$ electrode for two $H_{\|}$ values that are both higher than the switching threshold. Pulse width is 300 picoseconds.

Figure 2-15 above corresponds to a 900 Oe easy-axis pulse that is 300 ps long. The magnetization starts to switch, but then reverses back once the field over the electrode disappears. The plot on the left corresponds to a complete magnetization reversal for the same electrode, but with an 800 Oe easy-axis pulse.

In Figure 2-14, the location of the "glitches" as a function of easy-axis field and pulse width depends, among other factors, on the shape of the junction. This is because the overall three-dimensional magnetic potential is altered as the junction shape changes. Figure 2-16 shows the results of the same simulation on the "distorted" shape.

For a given junction size, the question of what pulse width will start to yield these glitches is strongly related to the damping parameter, α. This comes as no surprise, since the degree of damping determines the natural frequency of oscillations in a system (hence the time it takes for the system to settle down). The smaller the damping, the longer it takes for the magnetization to settle into a trajectory that leads to switching; hence, the longer the pulse has to be in order to reverse the magnetization. Figure 2-17 below illustrates this fact; the only difference

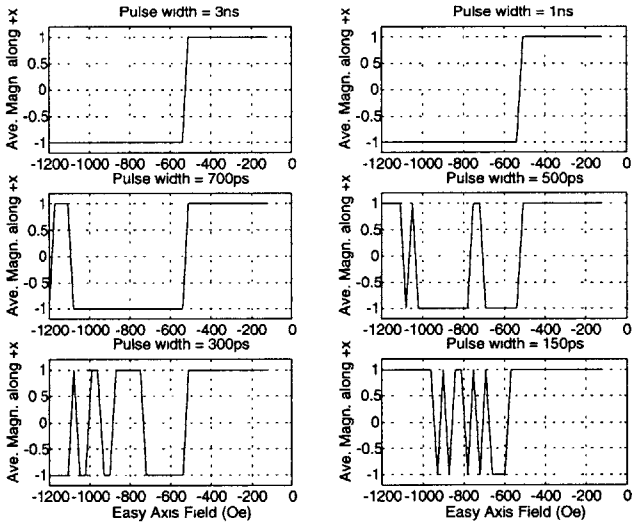

Figure 2-16: Pulsed magnetic field sweeps on the distorted shape for various pulse widths. The damping parameter is taken to be 0.01.

between these sets of simulations and those presented for the "nominal" shape is that here, α has been reduced from 0.01 to 0.003. Notice that the "glitches" now start occurring with even a 1 ns pulse. Also worth noting is that a 150 ps pulse is no longer able to reverse the magnetic moment at all.

For the sake of completeness, we present the results of the same type of simulations on a larger (0.28×0.84 μm^2) electrode. Figure 2-18 below show the "nominal" and "distorted" junction shapes. Figures 2-19 and 2-20 display the results of the pulse width studies on these two shapes.

It takes up to two days to simulate these relatively larger structures for a given pulse width. Due to time limitations, there is not as much pulse width variation as one would desire in our study with the larger "distorted" shape. However, it is clear from the plots that magnetization dynamics for short pulses is different for the two shapes.

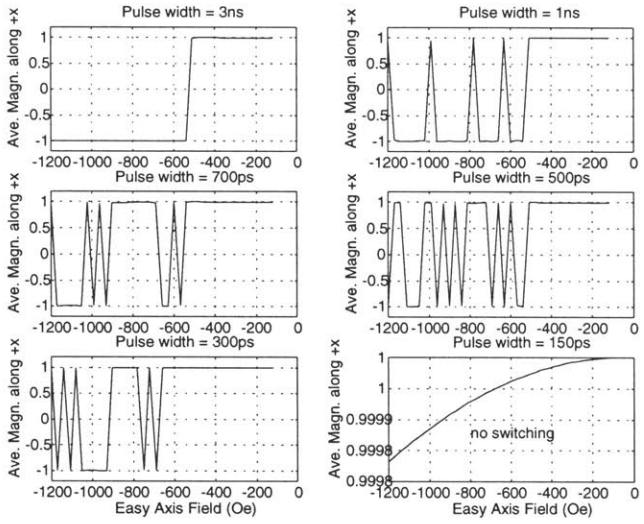

Figure 2-17: Pulsed magnetic field sweeps on the nominal shape for various pulse widths. The damping parameter is taken to be 0.003.

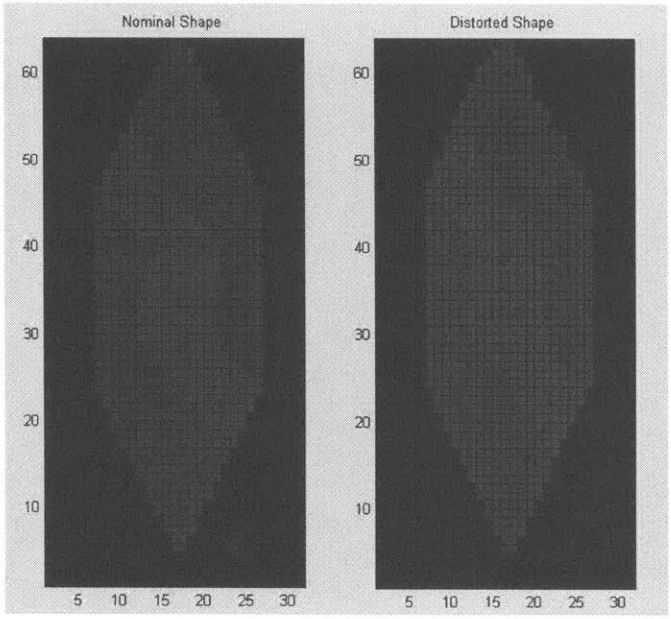

Figure 2-18: A comparison of two computational lattice shapes used in this study, for a 0.28×0.84 μm^2junction.

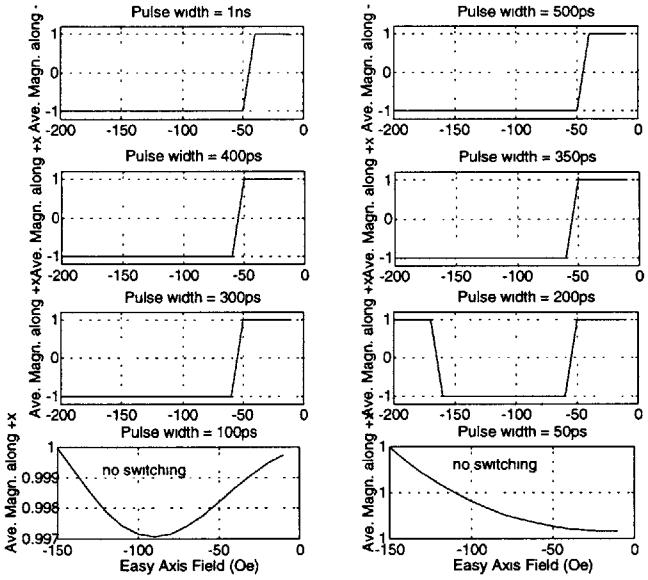

Figure 2-19: Pulsed magnetic field sweeps on the $0.28 \times 0.84\ \mu m^2$ junction (nominal shape) for various pulse widths. The damping parameter is taken to be 0.003.

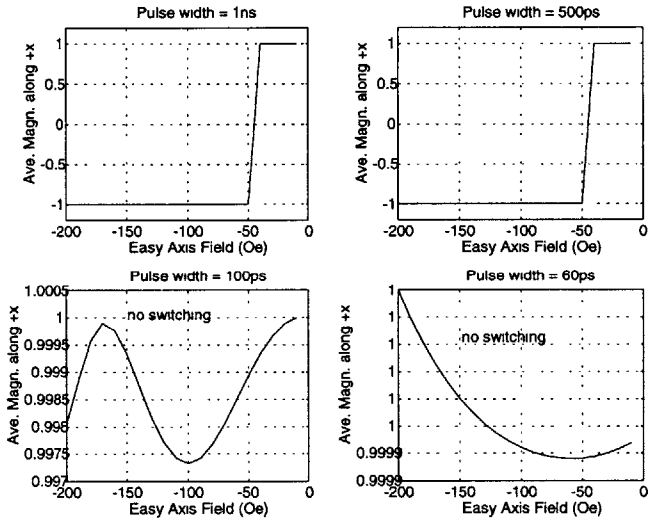

Figure 2-20: Pulsed magnetic field sweeps on the $0.28 \times 0.84\ \mu m^2$ junction (nominal shape) for various pulse widths. The damping parameter is taken to be 0.003.

Bibliography

[1] L.D.Landau, E.M. Lifshitz, and L.P. Pitaevski, *Electrodynamics of Continuous Media (2nd Edition)*, Pergamon Press (1984)

[2] M. Mansuripur, *Magnetization reversal dynamics in the media of magneto-optical recording*, J. Appl. Phys., **63**, 12 (1998)

[3] Allen H. Morrish, *The Physical Principles of Magnetism*; Sōshin Chikazumi, *Physics of Magnetism*, John Wiley & Sons, Inc. (1964); M. Cyrot, *Magnetism of Metals and Alloys* (1982)

[4] A. Gangulee and R.C. Taylor, J. Appl. Phys. **49**, 1762 (1978)

[5] M. Mansuripur and M.F. Ruane, IEEE Trans. Magn. **22**, 33 (1986)

[6] N.W. Ashcroft and N.D. Mermin, *Solid State Physics*, Saunders College Publishing. International Edition (1976)

[7] J.D.Jackson, *Classical Electrodynamics*, Wiley, New York (1975).

[8] W.H. Press, W.T.Vetterling, S.A. Teukolsky, B.P. Flannery, *Numerical Recipes in C*, Cambridge University Press, Second Edition (1995)

[9] Stoner, E.C., Wohlfarth, *Mechanism of Magnetic Hysterisis in Heterogeneous Alloys*, E.P., Trans. Roy. Soc. (London) **A240**, 599-644,

[10] Yu Lu, et. al., *Observation of magnetic switching in sub-micron magnetic-tunnel-junctions at low frequency*, J. Appl. Phys. (to be published)

Chapter 3

Experiments

3.1 Introduction

One of the factors that will determine the competitiveness of MRAM is the speed with which a bit can be written on a single memory element. Therefore, a detailed study of switching dynamics is necessary. In the previous chapter, we attempted to understand some aspects of magnetization dynamics in ferromagnetic thin films through computer simulations. Here, we will first introduce an experimental setup and method that has been used to characterize magnetization switching in metallic electrodes of certain ferromagnetic alloys. Later, we will present some results and attempt to explain certain characteristics in the light of the computer simulations.

3.2 Setup

Figure 3-1 below shows a simplified diagram of the experimental setup.

3.2.1 The Electromagnet subsystem

The most fundamental part of the setup is a DC electromagnet that is capable of providing a magnetic field in two independent, orthogonal directions in the plane of the sample. The electromagnet is comprised of a toroidal ferromagnetic core, and low resistance wires wound around it. With careful winding, the residual field in one orthogonal direction due to the

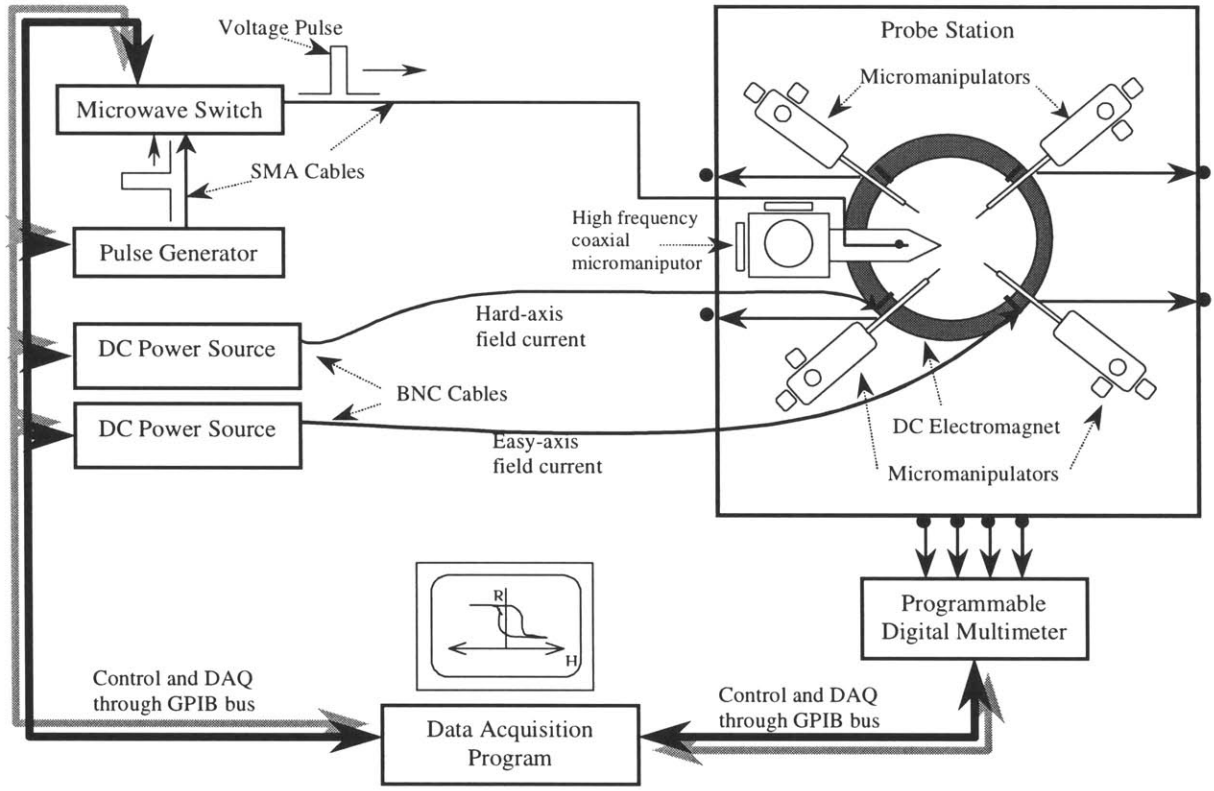

Figure 3-1: A schematic of the experimental setup.

presence of field in the other direction, can be made less than 1 %. The sample is aligned inside the magnet so that the orthogonal directions coincide with the easy and hard-axis directions of the junction electrodes. In this experiment, electromagnets that are capable of producing about 250 Oe safely in either direction have been used[1]. This value is large enough to switch the magnetization in the top electrode of the MTJ, yet smaller than the coercive field of the lower electrode. Hence, by monitoring the changes in the resistance of the junction, the average magnetization direction inside the top electrode can be studied.

Two bipolar power supplies (KEPKO BOP 50-8M-4882), capable of providing 8 Amperes of direct current, have been used to drive the electromagnet. They can be fully controlled by a computer through a General Peripheral Interface Bus (GPIB). Using a digital gaussmeter, the electromagnet subsystem is calibrated, and the calibration parameters are stored on computer

[1]The limit on the magnitude of the magnetic field produced is determined by heating on the magnet wires.

memory. The calibration procedure is repeated every time a new magnet is to be used.

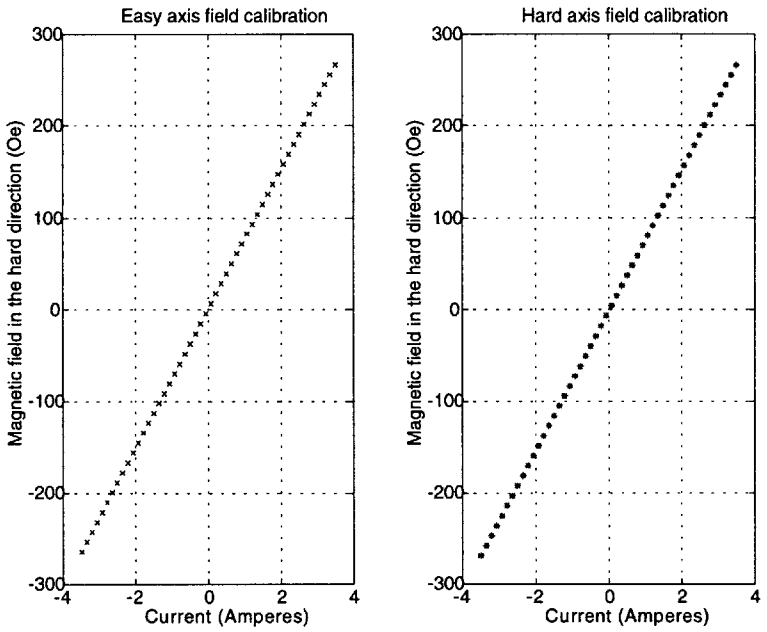

Figure 3-2: Calibration curves for the two orthogonal directions of the electromagnet.

3.2.2 The pulser subsystem

One of the main purposes of this experimental study is to characterize the switching dynamics of the junctions under possible memory configurations. This requires applying on-chip pulses with different widths and magnitutes. A "magnetic pulse" is created by a voltage pulse that propogates through a transmission line on the chip itself. Specifically for this study, high frequency chip sites with transmission lines to support several GHz have been designed and produced.

A 500 MHz pulse generator (Hewlett Packard 8131A) is used to create the necessary easy-axis field. The specifications for this particular instrument include pulses with a minimum of 200 picosecond rise-time, and about 5 % accuracy. The voltage pulse amplitude ranges from 100 mV to 5 Vp-p with a resolution of 10 mV. The 8131A is initialized and controlled fully

through the GPIB interface by the data acquisition program. The pulse generator is run at the External Trigger mode and at each measurement point, a single, isolated pulse is applied before a resistance measurement is taken.

The signal from the generator is fed through an SMA cable into a microwave switch (44476B Microwave Switch, Hewlett Packard 3488A Switch/Control Unit). This switch unit provides a 50 Ω path, more than 90 dB signal isolation, and it is capable of supporting signals up to 18 GHz without any appreciable loss. The need for this switch arises from the fact that some of the on-chip signal paths for pulses and resistance measurements coincide; hence the pulsing circuitry needs to be disconnected each time a resistance measurement is taken, and vice versa. Even though there exists a soft disconnect switch on the pulse generator, it produces undesired glitches every time it is engaged. It has also been observed that changing the height of the voltage pulse also produces small yet appreciable glitches that disturb the measurements. The microwave switch solves these problems, while still maintaining the high frequency characteristics of the transmission line.

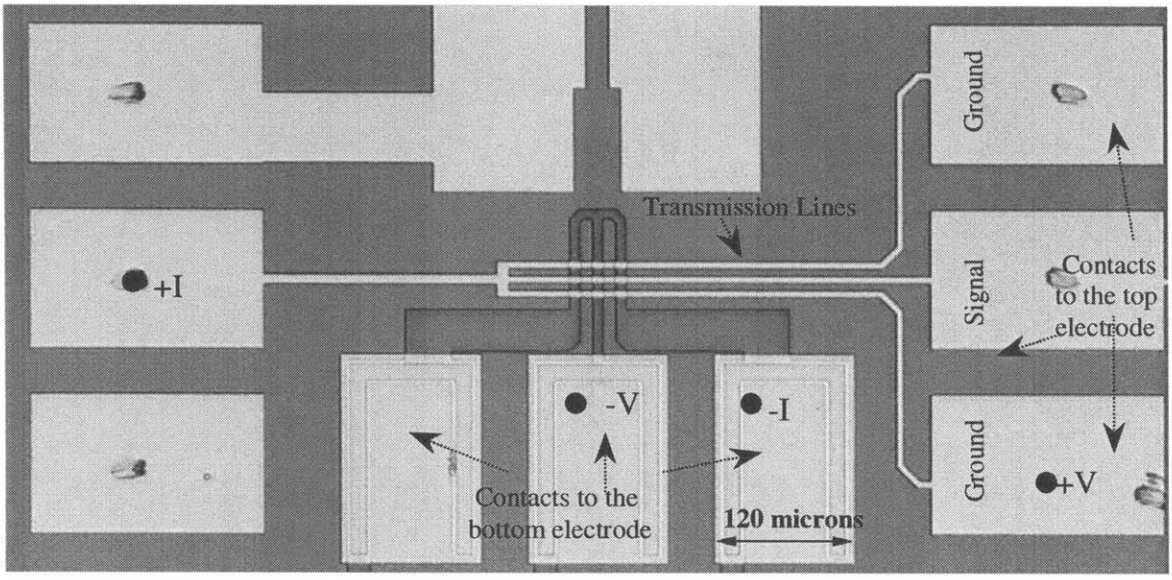

Figure 3-3: Photograph of an earlier test site design. The black dots mark the pads where probes land for four-point resistance measurements. The three pads on the right are used for pulse experiments.

The voltage pulse is transmitted onto the chip through a special probe (Cascade Microtech,

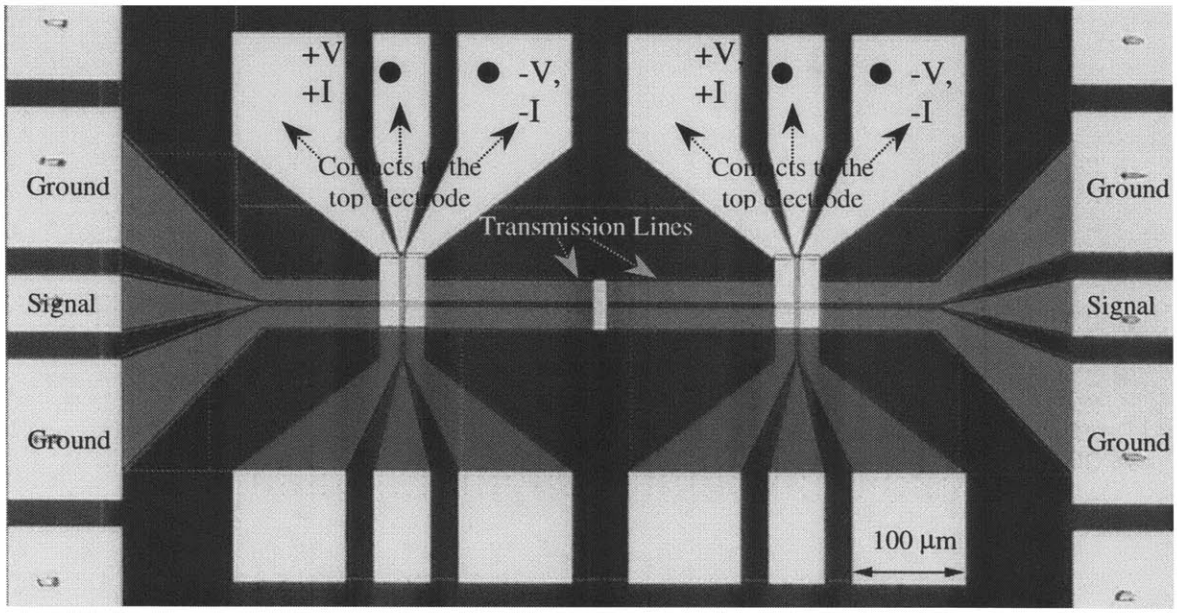

Figure 3-4: Photograph of a more recent test site. This design only allows two-point resistance measurement, but it fits more junctions per chip to be tested.

GSG Air Coplaner^{TM} Probe) on a microwave probe positioner (Cascade Microtech, 101-124). The positioner allows for accurate control of the high frequency probe in three dimensions, as well as its pitch with respect to the sample surface. The probe itself consists of three leads that contact the sample – the middle pin is the signal line, and the outer two pins define the ground. The 3 dB roll-off bandwidth of the probe is specified to be 18 GHz. Figure 3-3 and 3-4 show the two chip designs used in these experiments to characterize the high frequency switching of MTJs. Both of the layouts used for these chips have been designed to comply with the current experimental setup; hence exchanging the two chips requires almost no change to the measurement system.

In both cases, the center wire carries the pulse signal and is responsible for providing the magnetic pulse to switch the MTJ just underneath. The outer two wires form the return path of the signal. Notice that the symmetry of the wire setup allows for the reduction of stray fields along the easy axis due to current flowing in the outer wires, while minimizing the area covered by the on-chip transmission line, thereby reducing parasitic inductance associated with the circuitry. The width, length, thickness, and the material (mainly gold) of the lines on the

chip have been chosen to provide a 50 Ω termination; however, reflections due to impedance mismatch cannot be avoided. The current implementation is quite ambitious already and the approach is the first of its kind; however, for a complete study with "clean" pulses, a driver circuitry incorporated on the chip is necessary.

3.2.3 The probe station

The samples are placed at the center of the electromagnet, on the stage of a probe station (Micromanipulator Co., Model 8800) that allows computerized and manual (joystick) control of the sample location with respect to the manipulators in three dimensions. This provides easy access to the sample, as well as practical speed in changing the junction to be measured. We use a custom stage extension that fits inside the electromagnet and allows the sample to be positioned in the middle of the toroid in the vertical direction. The piece also extends the vacuum hold of the probe station, so the sample can be held firm in position during measurements by turning on the vacuum supply of the station.

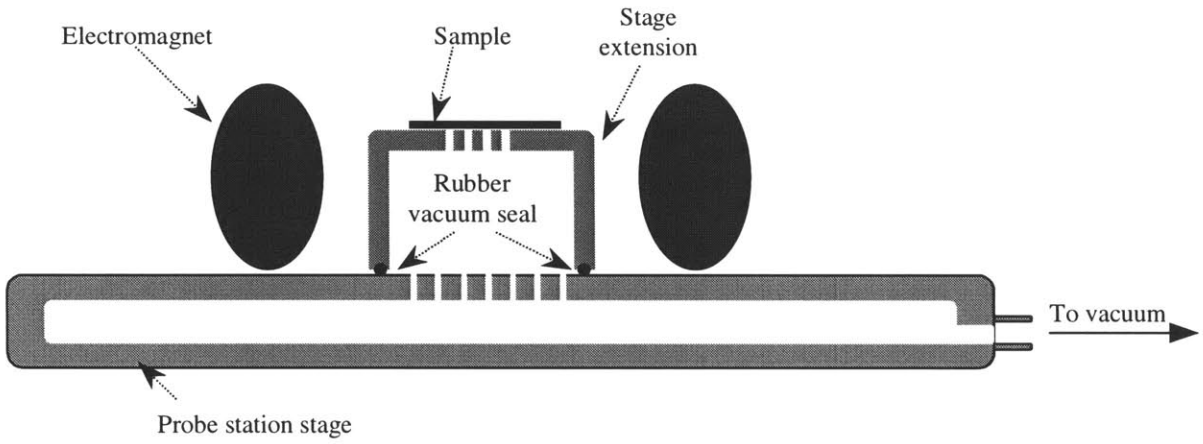

Figure 3-5: A cross-section sketch of the probe station stage.

3.2.4 The resistance measurement subsystem

This subsystem consists of four single probe micromanipulators (Micromanipulator Co., Model 450/360 MT), a digital sourcemeter (Keithley, Model 2400), and the associated wiring. The

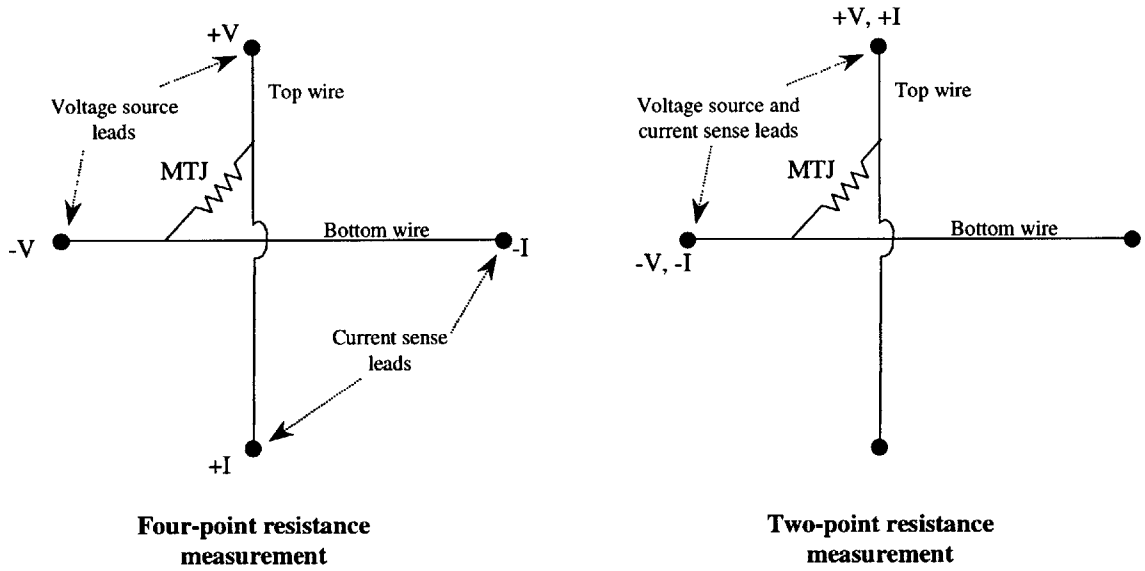

Figure 3-6: Difference between the two types of resistance measurements used. Notice that with a two-point measurement, the resistance seen by the sourcemeter includes the contribution from those sections of the wires along the signal path.

sourcemeter can be setup to perform either a four-point or a two-point resistance measurement, depending on the chip design. Figure 3-6 below illustrates the difference between the two types of measurements. It is important to point out that ideally, a four-point setup measures the resistance of the MTJ only, whereas a two-point measurement includes the series resistance of the wires as well.

A two-point resistance measurement, therefore, is not accurate in the absolute sense. However, it is only the relative change in magnetoresistance that we are after, hence, no information about the magnetization switching is lost with a two-point resistance measurement.

The sourcemeter is initialized and programmed completely by the computer controller program through the GPIB interface. For each resistance measurement, the device is triggered by the computer to take 20 consecutive readings, average them, and send the result back to the controller. Throughout the experiments, the sourcemeter has been programmed to act as a voltage source/current probe – applying 10 mV across the junction and measuring the current through it. The current protection threshold has been placed at 0.1 mA; hence, a measurement is terminated if the resistance seen is less than 10 Ω (all the functional MTJs produced have

resistances larger than 100 Ω), and the current protection feature prevents potential damage to the rest of the chip in case one particular junction is shorted out.

3.3 Procedure

In this work, we have attempted to characterize the magnetization switching in MTJs using through several steps using the experimental setup described above. The main experimental procedures can be categorized in four different contexts.

3.3.1 External Magnetic Field Sweep

Using the external electromagnet, the easy axis field is swept across the magnetoresistance saturation range, and the DC switching fields are obtained (see Figure 3-7 below). Further characterization is also possible by systematically sweeping the easy axis for different values of the hard axis field. A typical MRAM with a large matrix of MTJs would naturally address individual memory elements with two orthogonal magnetic fields. Hence, this study is crucial in determining the optimal field direction and strength that will be needed in a memory application.

3.3.2 Pulse Sweep Calibration

On-chip magnetic fields are produced by current pulses that travel across the micro-transmission lines above the junctions. Unfortunately, one cannot place a magnetic sensor right next to a junction and measure the magnetic field induced by the traveling pulse. Although it is possible to estimate numerically what the induced magnetic field strength is, uncertainties is processing conditions, variability of the contact resistance between the probe and the test site, etc. make such calculations inaccurate. Hence, a physical calibration of pulse height versus induced magnetic field is necessary. This is achieved by a "pulse sweep", in which the pulse height is slowly ramped up and down across the MR saturation range, in a similar manner as in the first step. The main difference between this procedure and the "DC field sweep" is that the magnetization of the top electrode is reset back with a full scale negative pulse in between

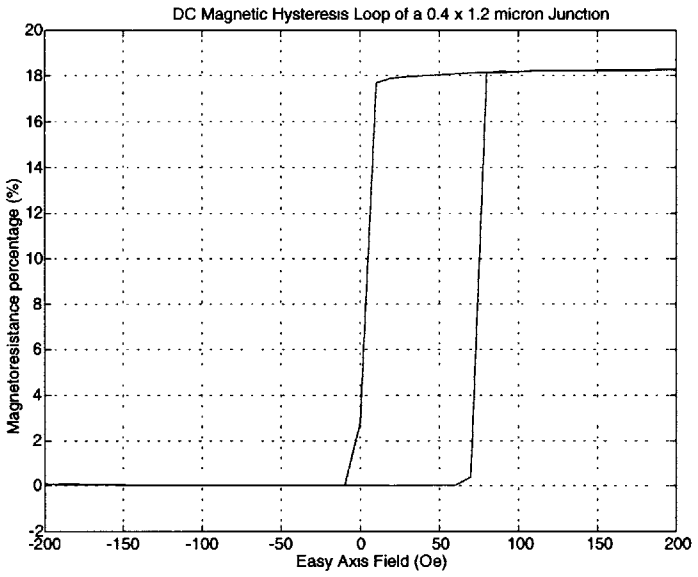

Figure 3-7: Magnetoresistance vs. the applied easy-axis field for a magnetic tunnel junction. Notice that the magnetic hysterisis loop creates two stable states at zero field.

each "sweep" pulse. Therefore, magnetization reversal, if it occurs, is always spontaneous and due to a single pulse.

The pulse width for this calibration is chosen relatively large – i.e., 2 milliseconds. The literature, as well as our earlier experiments and simulations show that magnetic switching in thin films of ferromagnetic alloys occurs on the order of nanoseconds, even picoseconds. Hence, a 2 ms pulse behaves like the limiting DC case, and the pulse heights that cause the device to switch states are then referred to the DC switching fields. This calibration procedure is performed every time a new transmission line needs to be used.

3.3.3 Pulse Sweep with Various Pulse Widths

Once the calibration between pulse height and induced magnetic field is achieved, we proceed to study the effects of varying the applied pulse width. Pulse sweeps with widths ranging from 100 nanoseconds down to 2 ns are performed and the corresponding switching fields are noted. Figure 3-9 below presents a sample sweep with a pulse width of 100 nanoseconds.

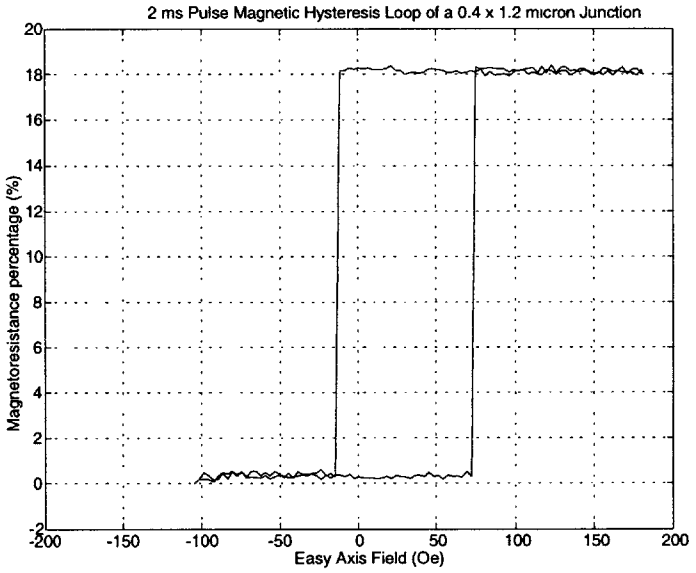

Figure 3-8: Magnetoresistance vs. applied easy-axis magnetic pulse. The field values where switching occurs are calibrated with respect to the thresholds in the DC sweep.

Preliminary studies indicated that the switching field thresholds increase as pulse widths become narrower (see Figure 3-10). This is expected, since a certain amount of energy is required to reverse the magnetization of the top electrode in a MTJ, and the energy delivered to the electrode is roughly proportional (assuming no-loss transmission) to the area under the pulse. This phenomenon implies that certain design factors, such as the maximum applicable voltage inside a MRAM chip, might ultimately determine the speed of the memory devices. The goal of this particular study of pulse widths and switching thresholds is to provide more experimental data on, and a deeper understanding of, this phenomenon.

3.3.4 Directional Characterization and Switching Statistics

In order to understand better how the magnetization of an electrode responds to applied magnetic field pulses, we characterize the magnetic "loops" of the junctions for various hard axis fields. It is known that magnetization reversal in a uniaxial ferromagnetic thin film is a function

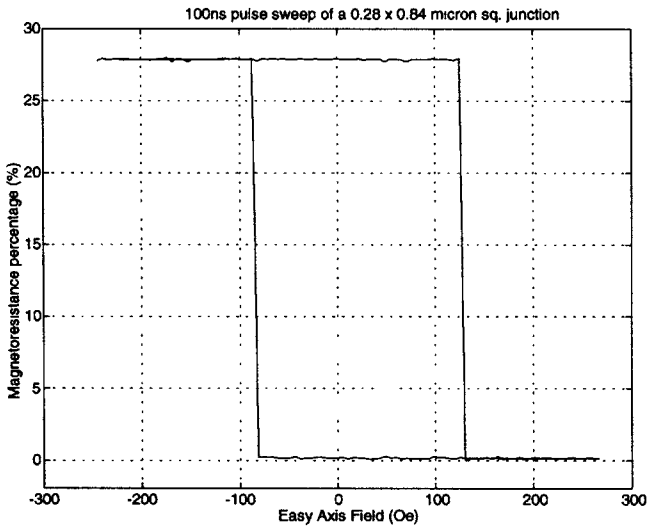

Figure 3-9: MR vs. magnetic pulse magnitude in a typical pulse sweep of a MTJ. The applied pulse width is 100 nanoseconds.

of the direction of the applied field. In particular, fields with a hard-axis component can reverse the magnetization more easily than those fields aligned with the easy axis of the film. This is due mostly to the ferromagnetic film's uniaxial anisotropy energy, which introduces an angular dependence to the total energy term.

We have also investigated the distribution of switching fields by taking repeated measurements. With enough data to provide significant statistics, this particular study should reveal the shape of the effective potential between the two stable states of the ferromagnetic film. Understanding the potential, in turn, should inform us more about the dependence of operating conditions on several processing parameters. Moreover, such statistics will help designers choose comfortable settings– e.g., the minimum required easy-axis pulse to guarantee switching all the time. The data acquired for the study reported here is not sufficient to determine an effective switching potential empirically, yet it is sufficient to present a general picture for memory designers to revise their layouts.

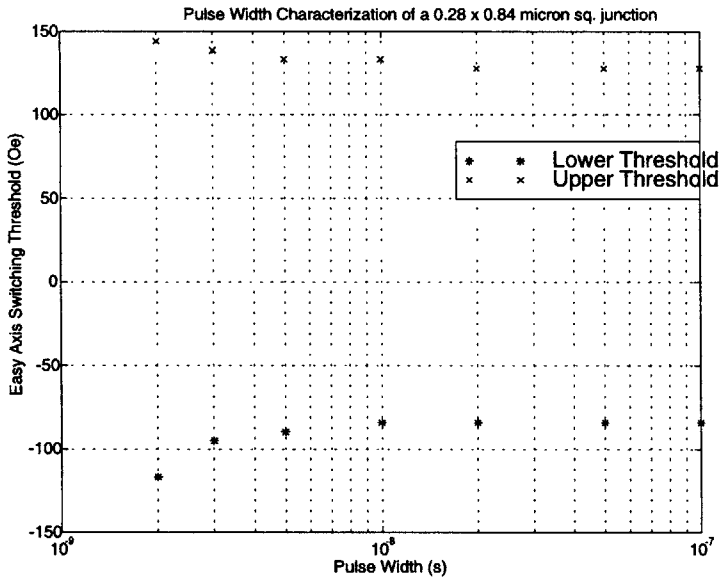

Figure 3-10: Switching field vs. pulse width in a junction. The general trend is an increase in the switching field threshold as faster pulses are used.

3.4 Results

The first two procedures described above are used mainly for the calibration of a test structure in preparation for the latter two steps of the experiment. In this section, we will discuss some results from pulsed experiments in which switching characteristics have been studied as a function of the hard-axis field applied, as well as the width of the pulse itself.

3.4.1 Directional Switching Characteristics

One lesson learned from experiment is that the magnetization reversal process is different in the dynamic case (i.e., with pulses) and the DC case (with the electromagnet). The "asteroid" studies performed with pulsed easy-axis fields reveal this distinction clearly. Figure 3-11 below presents the comparison for one junction.

Multiple points for a given hard-axis field on the plot at the bottom of Figure 3-11 represent the distribution of switching thresholds. Notice that the overall shapes of the two "asteroids" shown do not match. In fact, it was observed that none of the "pulse asteroids" acquired on the

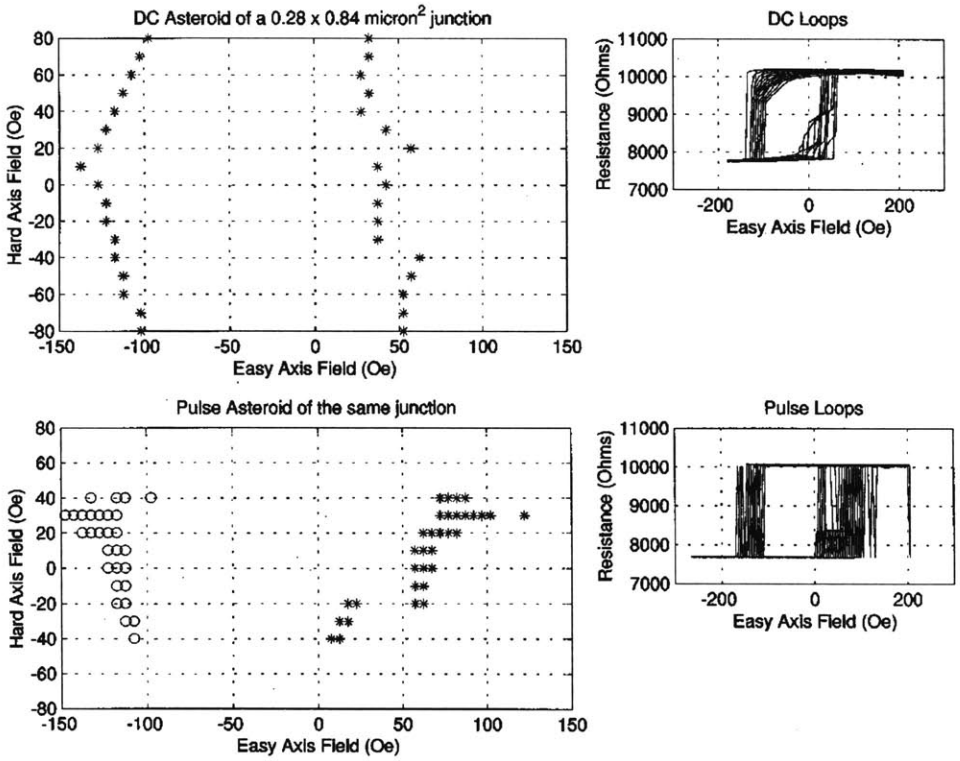

Figure 3-11: A comparison of the dependence of magnetization reversal on applied field direction in the static (DC) and dynamic (pulsed) cases. The pulse width used is 10 nanoseconds

high speed test sites of either kind matched the corresponding data in the DC measurements. This fact is also supported by the evidence from the simulations, suggesting that the dynamics involved in magnetization reversal with short width pulses is fundamentally different and more complex than the static case.

Figure 3-12 below shows a histogram plot of pulsed asteroid data for the same junction as in the previous figure above. Denser lines correspond to a larger frequency at that particular histogram bin.

For this particular junction, there is a significant spread in the threshold values. This distribution is partly a function of the finite rise-time and the limited accuracy of the pulse generator and the frequency response of the entire signal path for the pulses. However, it is mainly affected by the wire geometry, the junction size and shape, as well as the material of the

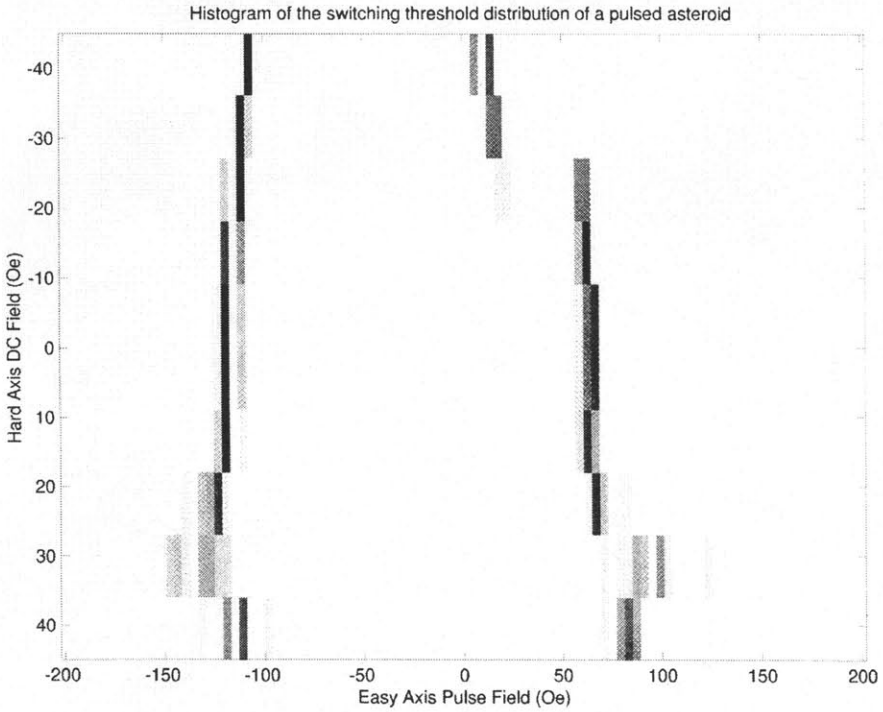

Figure 3-12: The distribution of the switching thresholds as a function of easy and hard-axis fields, for the same device as shown in the earlier figure. The pulse width used is 10 ns.

top electrode (i.e., the free layer). The nonuniformity of the magnetic field induced under the micro-transmission line is also responsible for the magnitude of the spread. Unfortunately, at this stage, there simply is not enough data to provide the statistics needed to make an informed judgement about the contribution of each of these effects to the phenomenon observed.

One thing we can study here, though, is the width of the voltage pulse applied and the effect it has on the spread of the thresholds. Figure 3-13 shows one such study, where pulse asteroid measurements have been taken using a $0.28x0.84$ micron^2 junction (same size and material as the previous example).

Notice that the overall shape of the asteroids is the same, but there is variation in the distribution from point to point. It is interesting to note visually that the spread associated with the lower threshold is less than the scatter of the higher threshold. In fact, one can

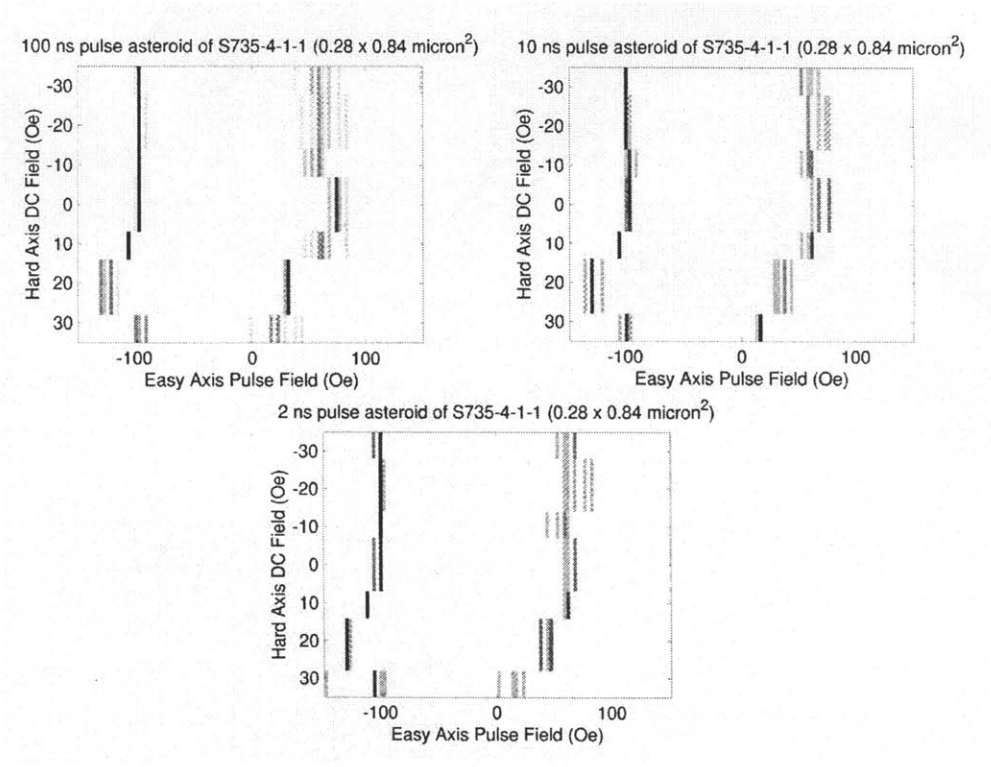

Figure 3-13: Pulse asteroid histograms on a MTJ at successively smaller pulse widths.

quantize this by taking the average standard deviation of the distributions for both the lower and the higher switching values. Table 3.1 below displays the results.

One other information Table 3.1 above supplies is that the statistics do not change appreciably for different pulse widths. This result indicates that the spread is primarily caused by effects that are independent of frequency, such as the magnetization dynamics and the design of the test site, as opposed to the frequency response of the signal path for pulses.

	100 ns	**10 ns**	**2 ns**
$\bar{\sigma}_x$ **for lower threshold**	2.06	2.51	2.07
$\bar{\sigma}_x$ **for higher threshold**	5.88	5.25	5.95

Table 3.1: The average standard deviations for the two thresholds of the hysterisis loop. Each standard deviation value was calculated based on the distribution of a single threshold value for a given hard-axis field.

3.4.2 Pulse Width Studies

One thing that depends on the applied pulse width is the overall hysterisis loop of the free layer. It has been observed without exception that shorter pulses result in larger hysterisis loops for a given device. The level of this dependence on pulse width varies from junction to junction. For most cases (see the results from a particular device depicted in Figures 3-14 and 3-15 below), the minimum width the pulse generator can supply – i.e., 2 ns – is not sufficient to study the really interesting range of values. Since most of the magnetization dynamics takes place within the first nanosecond or less, a faster pulse generator is necessary to better characterize the dependence.

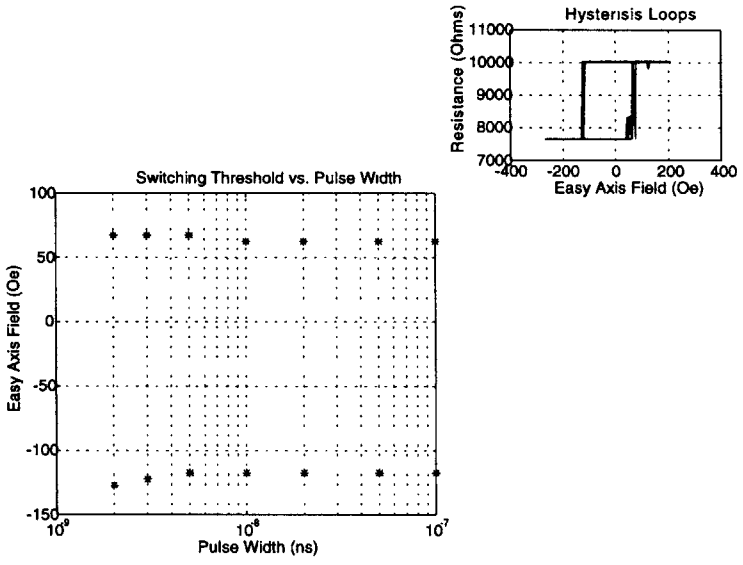

Figure 3-14: The switching thresholds of a $0.28x0.84$ micron^2 junction as a function of magnetic pulse width. The small plot at the top shows the individual hysterisis loops for each data point.

Despite the relatively slow HP 8131A pulse generator, the data acquired still reveals some interesting features. For example, a thorough analysis indicates that magnetization reversal of the free layer from the parallel to the antiparallel state (i.e., from low to high resistance) is more sensitive to the width of the magnetic pulse than the opposite transition is. Figure 3-16 below shows data from two different junctions to illustrate this point.

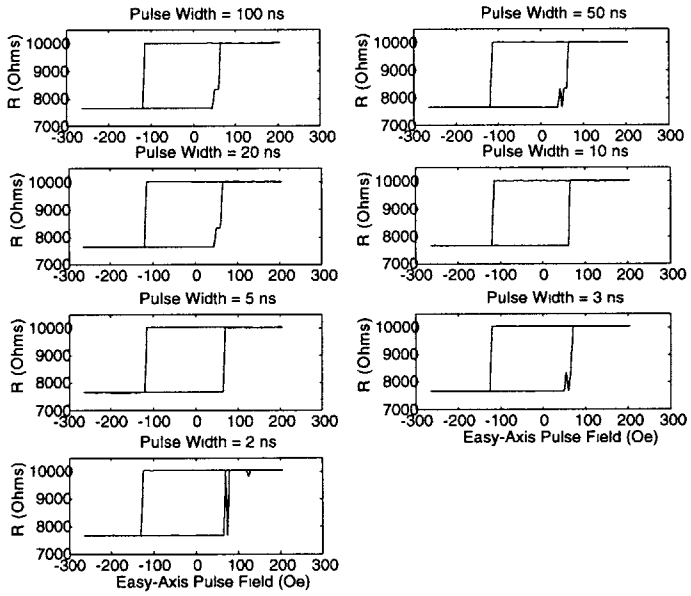

Figure 3-15: Individual hysterisis loops of the $0.28x0.84$ μm^2 junction with various magnetic pulse widths.

The data suggests that the energy surface as seen by the average magnetization vector of the free layer is not symmetric about $\phi = 90^o$, as it can be expected from the form of the uniaxial anisotropy energy term. Evidently, the high R (resistance) state is less stable than the low R state, an observation that might have significant implications for disturbance rejection and bit storage issues in a memory array composed of MTJs. This observation is further supported by the individual pulse hysterisis loops of many junctions, one of which is shown in Figure 3-17 below.

Notice from the figure that for certain values of the easy-axis pulse above the higher coercive field, the magnetization of the free layer does not switch from the parallel to the anti-parallel state. However, same field values in the opposite direction do not cause "glitches" in the antiparallel-to-parallel transition. This finding is totally unexpected in the light of our earlier studies, in which experimental characterization of MTJs were solely performed with DC fields and no instability of the high R state was observed. Apparently, this phenomenon is directly associated with the dynamics of the magnetization reversal, and it tends to appear only with a

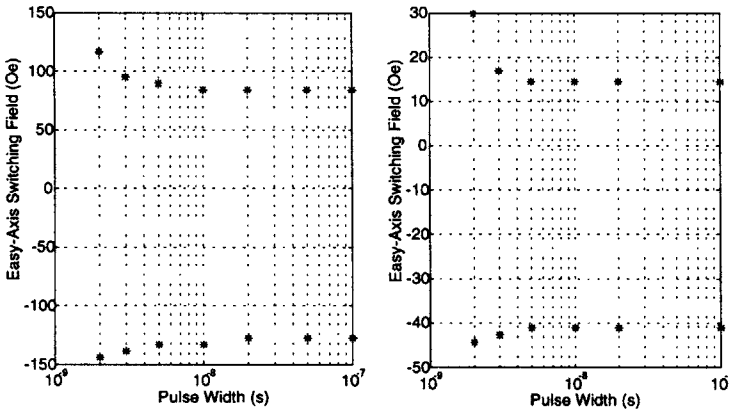

Figure 3-16: Dependence of the switching threshold on the applied pulse width is stronger for the low-high resistance switch (i.e., from parallel to antiparallel average magnetization orientation of the free layer).

short duration magnetic field pulse.

We would like to suggest a possibly deeper insight on this phenomenon. We have observed that the "glitches" seen in Figure 3-17 (and the data from the rest of the junctions tested) show a systematic dependence on the applied pulse width. It appears that as the pulse width gets shorter, those "glitches" get displaced as a function of easy-axis field, and more start to occur at different fields. As the pulse width gets even shorter, the "glitches" begin to merge, until they finally unite with the hysterisis loop. This behavior is exactly the type of phenomenon we presented in the previous chapter. The pulse width that begins to cause these "glitches" is determined strongly by the ferromagnetic resonance frequency of the free layer, which in turn depends on the damping parameter, α. The damping parameter is a material property. For a given pulse width, where these "glitches" occur is determined by the three dimensional path of the total magnetization vector, which is, among other factors, a function of the size and the shape of the free layer. Our earlier results with the simulations revealed that even small shape mismatches lead to characteristic differences

Figure 3-18 below illustrates this behavior for a different MTJ with the same free layer material and same nominal size and shape as the junction in the previous figure (although pro-

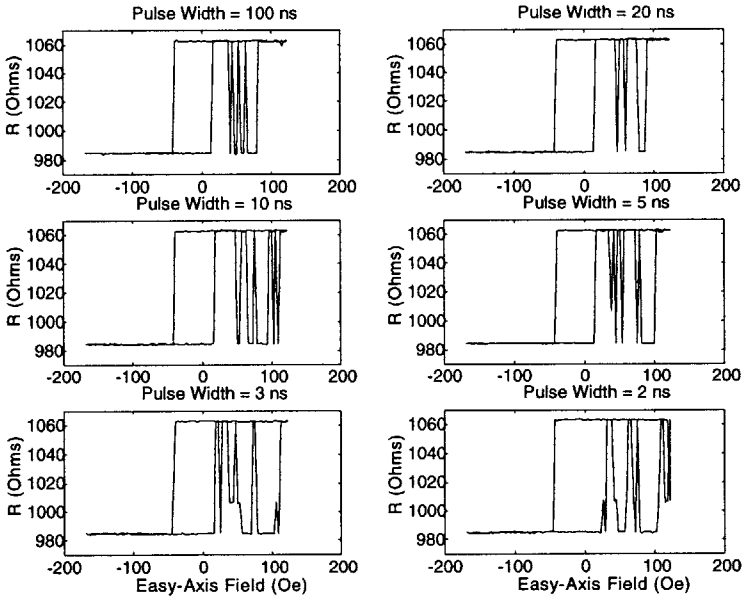

Figure 3-17: Individual hysterisis loops for a $0.28x0.84$ micron^2 junction for different values of the applied pulse width.

cessing techniques result in some shape variation in these sub-micron devices). Notice that this MTJ is partially shorted out (relatively low resistance); yet it shows enough magnetoresistance to be studied effectively.

We believe this phenomenon is strongly related to what has been observed in our simulation studies, where we have demonstrated the same kind of systematic dependence on the applied pulse width. The experimental conditions are significantly more complex than those assumed during the simulation[2]; yet, the similarity between these sets of simulation and the experiment results is astonishing.

An important point here is that the "glitches" in the experimental hysterisis loops start occuring for pulse widths larger than those deduced from the simulation data. Using the insight

[2]Some of the assumptions adopted for simplicity during the simulations are: infinite rise time for applied pulses, uniform magnetic pulse field throughout the free layer, uniform local magnetization magnitude over the entire ferromagnetic material, neglect of magnetic coupling to the pinned layer, as well as magnetostriction and temperature effects. None of these assumptions are valid for the experimental study.

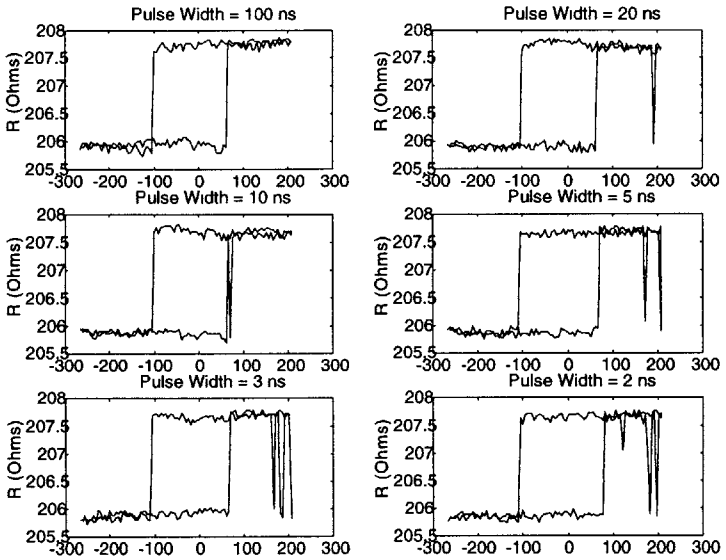

Figure 3-18: Individual hysterisis loops of another 0.28x0.84 micron^2 junction for various pulse widths

gained from the simulations and the dependence of these glitches on the damping parameter, we may infer that the α associated with the MTJs studied in this work is probably smaller than what has recently been suggested in the literature [1] (i.e., about 0.01). At the time of this writing, new work is under way in our group to measure the value of the damping parameter in a completely different experimental setup. Since the precise value of α eventually determines the switching speed of the MTJs (and hence, the speed of MRAM devices), it is one of the most significant factors that will determine the success of the memory devices built from these junctions.

The challenge of producing faster MTJs will require new research and development in materials science and fabrication technology. So far, approaches to systematically vary α are almost completely unexplored. It appears that there is still some time before micromagnetics can even begin to achieve the speed and the consistency one needs in order to use them in MRAM applications.

Bibliography

[1] R. H. Koch et. al., *Magnetization Reversal in Micron-Sized Magnetic Thin Films*, Phys. Rev. Lett., **81**, 4512 (1998)

Chapter 4

Conclusion

In this thesis, we have investigated the dynamics of magnetization reversal in ferromagnetic transition metals and alloys. Magnetoresistance due to spin polarized tunneling through a ferromagnet-insulator-ferromagnet structure has been used as the observable macroscopic variable to characterize magnetic switching in these materials.

In order to study the effects of several parameters on magnetic switching, we have developed a micromagnetics simulation program. We have concentrated on a material with magnetic properties quite close to those of permalloy. This is not only because permalloy is one of the main electrode materials used in the fabrication of MTJs in this study, but also because it's been shown to display characteristics – such as a high magnetoresistance – that are desirable for a possible MRAM implementation. In particular, we have studied the effects of systematically varying the degree of magnetic damping in this material. It has been shown, through the switching threshold asteroid curves, that the damping parameter determines the symmetry of directional switching characteristics. We have also conducted "pulsed field" simulations, where the external magnetic field is introduced as a short duration pulse – similar to what is intended to happen in MRAM architectures. We have seen that the damping parameter also determines just how fast the free layer of an MTJ will reverse its magnetization under the influence of an external field. In fact, the entire switching dynamics of the junctions are closely coupled to this parameter. Future fabrication efforts will need to achieve good control on the damping parameter if fast, clean and highly reproducible magnetization switching characteristics are to be attained at all. The exact shape of the junctions is also an important factor that effects

magnetic switching, especially in very underdamped magnetic systems such as permalloy. It appears that more research and development on lithograpy techniques to yield better shape definitions will be worthwhile.

As part of our study on magnetization reversal dynamics, we have also designed high-speed sites on our test wafers. A complete experimental setup to observe and study the dynamics on these sites has been put together. Results from systematic studies with short pulses reveal that there is considerable variation in the switching thresholds as a function of the hard-axis field applied. More studies with different materials and MTJ configurations need to be conducted to improve the consistency of switching. It has also been observed that "glitches" in magnetic hysterisis loops do really occur in experiment, as well as the simulations. What comes as a surprise, though, is that these "glitches" start to occur for pulse widths much larger than those predicted in simulations that assume the currently accepted value of the damping parameter. This is an indication that in most of the junctions measured, the damping parameter is less than expected (i.e., less than about 0.01).

Our results from both the simulations and experiments strongly suggest that more research needs to be conducted to understand and control the damping parameter. Therefore, among our list of suggested future work, an experimental study on directly measuring the value of the damping parameter for different ferromagnetic transition metals and alloys comes first. We also think more computer simulations that concentrate on systematic variations of other magnetic properties may help determine the most suitable material to be used as junction electrodes. Moreover, there exists no clear, systematic experimental data on the effects (over magnetization switching thresholds) of pulsing both the easy and the hard-axis fields simultaneously. Hence, we believe it is extremely important to characterize the magnetization dynamics with bidirectional pulses, thereby understanding the issues that will arise in a MRAM architecture (where orthogonal word and bit lines will carry short pulses to write bits). Of course, using CMOS circuitry to drive the easy and hard-axis lines on the chip itself will be the ultimate test.

It is only quite recently that significant magnetoresistance values have been achieved in spin-polarized tunneling experiments. The quest to use magnetic tunneling junctions in a random access memory is very young, yet quite promising. It is our hope that this thesis work will stimulate more research on these devices, and their possible use for memory applications.

Acknowledgements

The author would like to take this opportunity to thank all the members of the MRAM group at IBM T. J. Watson Research Center. Special thanks go to William Gallagher for his invaluable support and enthusiasm, without which this thesis would not be possible. Also, for their help and useful discussions about experiments and simulations, Yu Lu, Philip Trouilloud and Roger Koch deserve credit. I would like to thank David Abraham for his collaboration during the simulations, and Makhlouf Redjdal from Boston University for his invaluable assistance in the calculation of geometrical factors. Thanks to Jonathan Sun for the helpful discussions on physics, and John Connolly for the simulation time on his computer. I deeply thank Terry Orlando for always sparing the time to guide me through all stages of this work. Finally, I am grateful to my Mom and Dad for their never-ending support and love.

Appendix A

Integrating the LLG equation

In this section, we will discuss the issues involved in integrating the Landau-Lifshitz-Gilbert equation numerically. The discussion below is general and pertains to any first degree ordinary differential equation (O.D.E.) of the form $y' = f(x, y)$ with $y(0) = y_0$ as the initial condition. A numerical approach to solving an O.D.E. involves discretizing the functional form at certain time steps, so that the computer actually solves a difference equation.

The LLG equation is a stiff ordinary differential equation. Primitive integration methods – such as Euler's method, the trapezoidal method, or the generalized Simpson's rule – are mostly time-inefficient, in that they all require too small integration steps for the computation to be practical. Efficiency and accuracy are better captured in multistep methods that use information from past values to compute the next value of the function at the next time step. Most generally, a multistep formula with $p + 1$ steps is given by

$$y_{k+1} = \sum_{i=0}^{p} a_i y_{k-i} + h \sum_{i=-1}^{p} b_i y'_{k-i} \tag{A.1}$$

Here, h is the integration step length. In a computer algorithm that adapts a multistep formula, the values y_{k-1} and y'_{k-1} are saved in a First In First Out (FIFO) buffer as they are computed. Notice that getting such a calculation started is tricky, since for $k < p$, there are not enough past values to calculate y_{k+1}. Generally, a multistep algorithm starts with a single step

formula using a very small time interval; once the first p values are computed, the multistep formula takes over [1].

Depending on the desired accuracy and the particular method, the parameters a_i and b_i can be found through the method of undetermined coefficients. Notice that if b_{-1} is non-zero, then Equation A.1 becomes a non-linear equation, since it contains y'_{k+1} on the right and y_{k+1} on the left. A multistep formula with $b_{-1} \neq 0$ is called *implicit*, otherwise, it is *explicit*. A numerical integration method that is implicit is inherently more stable than an explicit method. However, this stability comes at the price of computational complexity due to the non-linear equations that need to be solved in an implicit method.

This problem can be remedied if one uses an explicit multistep formula to get an estimate of y_{k+1} first. Then, using $y'_{k+1} = f(x_{k+1}, y_{k+1})$, an estimate of y'_{k+1} can be substituted into the implicit formula. If the integration step h is small enough, several iterations of the implicit formula then converges on the true value of y_{k+1}[1]. This approach is known as the **predictor-corrector** method – the explicit formula is the predictor and the implicit equation is the corrector.

A particularly simple predictor-corrector method – namely, the Milne's method – has been used in this study to integrate the LLG equation[2]. The local error in the formulas

$$\begin{aligned} y^{(0)}_{k+1} &= y_k + \frac{4h}{3}\left(2y'_k - y'_{k-1} + 2y'_{k-2}\right) \qquad \text{(predictor)} \\ y^{(j)}_{k+1} &= y_{k-1} + \frac{h}{3}\left(y'^{(j-1)}_{k+1} + 4y'_k + y'_{k-1}\right) \quad \text{(corrector)} \end{aligned} \tag{A.2}$$

is $O(h^4)$. Notice that the difference in the results of the predictor and the corrector formulas is also a good estimate of the local error in the calculation. Indeed, the simulation program uses this difference to control the integration accuracy of the LLG equation.

The integration starts with a trapezoidal method using a very small time step. Once the values for y'_0, y'_1, and y'_2 are calculated, Milne's method takes over, using a relatively large and

[1] A fixed number of iterations eventually leads to divergence of the solution. Newton's method is usually the most prefered way to achieve convergence, although it is computationally intense. See [1] for mathematical details on issues of convergence for a multistep method.

[2] See [1] for a derivation of Milne's method, as well as a discussion on local errors involved.

practical time interval. At each step, the predictor and corrector formulas are applied once, and the corrector is then simply iterated until $y_{k+1}^{(j)} - y_{k+1}^{(j-1)}$ becomes smaller than a predefined tolerance level (usually, 10^{-4}). Unfortunately, simplicity comes at a price: self-iteration of the corrector formula is not sufficient to keep the fourth-order error terms under control. During a long integration, these error terms eventually accumulate and create oscillatory behavior in the solution.

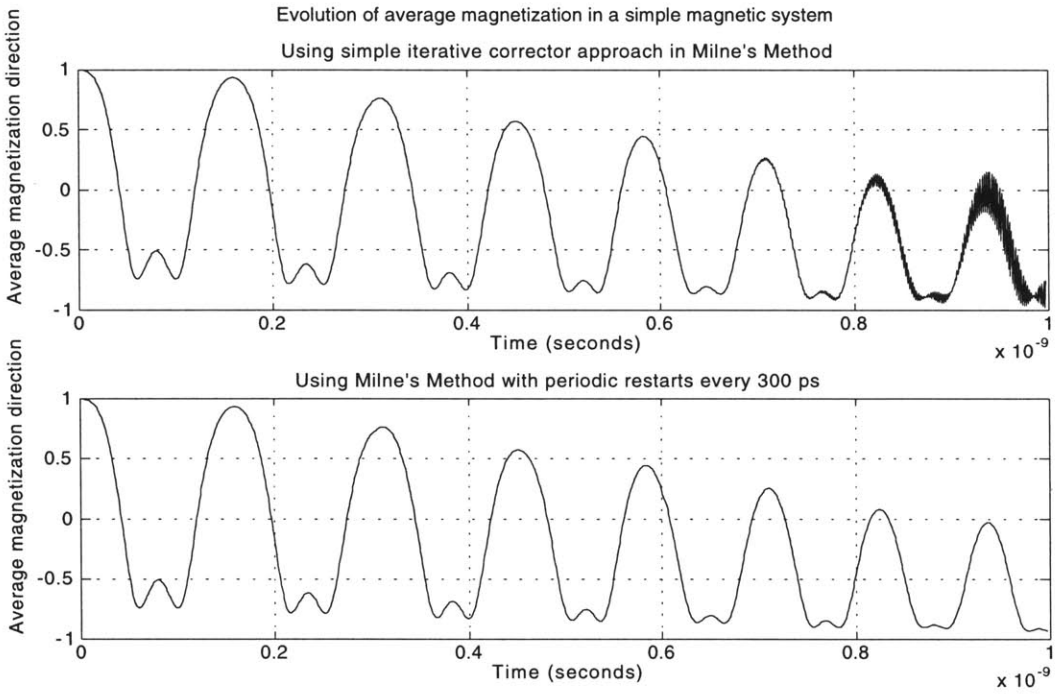

Figure A-1: Comparison of the stability properties of integration with Milne's method. The plots show the evolution of the average magnetization in the easy-axis direction of a simple ferromagnetic film.

In order to remedy this problem, the calculation is periodically restarted – after a certain period, Milne's method gives way to the trapezoidal method for a short interval in which a tiny time step is used to guarantee stability. This way is a great deal more practical and efficient than iterating with Newton's method, since the latter involves evaluating the Jacobian (hence, the right-hand side) of the LLG equation, which is computationally rather costly. Restarting Milne's method periodically guarantees the stability of a very long integration, while keeping

the calculation fast. See Figure A-1 for a comparison of results with and without periodic restarts.

Bibliography

[1] Numerical Methods, Software, and Analysis. John R. Rice. McGraw-Hill Book Company, 1983.

[2] R. Butler, E. Kerr, *An Introduction to Numerical Methods*, Pitman Publishing Corp. (1962)

[3] Numerical Recipes in C. W.H. Press, W.T.Vetterling, S.A. Teukolsky, B.P. Flannery. Cambridge University Press. Second Edition. 1995

Appendix B

Calculating the Demagnetization Kernel Coefficients

The demagnetization kernel is introduced in Chapter 2 as

$$G'_{ij}(\mathbf{r}) = \frac{3\hat{r}_i\hat{r}_j - \delta_{ij}}{r^3} \quad ; \quad \mathbf{r} = \mathbf{x} - \mathbf{x}_1 \quad \text{and } i,j = x,y,z$$

and the associated kernel coefficients are defined as

$$G_{ij}(\mathbf{x}) = \int_{-h/2}^{h/2} dz_1 \int_{-d/2}^{d/2} dy_1 \int_{-d/2}^{d/2} dx_1 G'_{ij}(\mathbf{x} - \mathbf{x}_1)$$

so that the Fourier transforms of the demagnetization field components are given by

$$\begin{aligned}
\tilde{H}_x(\mathbf{k}) &= M\left(\tilde{G}_{xx}(\mathbf{k}) * \tilde{m}_x(\mathbf{k}) + \tilde{G}_{xy}(\mathbf{k}) * \tilde{m}_y(\mathbf{k}) + \tilde{G}_{xz}(\mathbf{k}) * \tilde{m}_z(\mathbf{k})\right) \\
\tilde{H}_y(\mathbf{k}) &= M\left(\tilde{G}_{yx}(\mathbf{k}) * \tilde{m}_x(\mathbf{k}) + \tilde{G}_{yy}(\mathbf{k}) * \tilde{m}_y(\mathbf{k}) + \tilde{G}_{yz}(\mathbf{k}) * \tilde{m}_z(\mathbf{k})\right) \\
\tilde{H}_z(\mathbf{k}) &= M\left(\tilde{G}_{zx}(\mathbf{k}) * \tilde{m}_x(\mathbf{k}) + \tilde{G}_{zy}(\mathbf{k}) * \tilde{m}_y(\mathbf{k}) + \tilde{G}_{zz}(\mathbf{k}) * \tilde{m}_z(\mathbf{k})\right)
\end{aligned} \tag{B.1}$$

The kernel coefficients need to be evaluated in real space before their Fourier transforms

can be inserted into equation B.1 above. In this appendix, we will present the calculation of one kernel coefficient, and list the results for the remaining coefficients.

Consider,

$$G_{xx}(\mathbf{x}) = \int_{-h/2}^{h/2} dz_1 \int_{-d/2}^{d/2} dy_1 \int_{-d/2}^{d/2} dx_1 \frac{3\hat{r}_x\hat{r}_x - \delta_{xx}}{r^3}$$

where

$$\hat{r}_x = \frac{|x - x_1|}{r} \Rightarrow \hat{r}_x^2 = \frac{(x-x_1)^2}{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2}$$

Then, we have

$$G_{xx}(\mathbf{x}) = \int_{-h/2}^{h/2} dz_1 \int_{-d/2}^{d/2} dy_1 \int_{-d/2}^{d/2} dx_1 \frac{3(x-x_1)^2 - \left((x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2\right)}{r^5}$$

With the substitution $c_x = x - x_1$, $c_y = y - y_1$, and $c_z = z - z_1$, we get

$$\begin{aligned} G_{xx}(x,y,z) &= \int_{z+h/2}^{z-h/2} dc_x \int_{y+d/2}^{y-d/2} dc_y \int_{x+d/2}^{x-d/2} dc_z \frac{3c_x^2}{\left(c_x^2+c_y^2+c_z^2\right)^{\frac{5}{2}}} - \frac{1}{\left(c_x^2+c_y^2+c_z^2\right)^{\frac{3}{2}}} \\ &= \arctan\left(\frac{c_y c_z}{c_x\sqrt{c_x^2+c_y^2+c_z^2}}\right)\Bigg|_{c_x=x+d/2}^{c_x=x-d/2}\Bigg|_{c_y=y+d/2}^{c_y=y-d/2}\Bigg|_{c_z=z+h/2}^{c_z=z-h/2} \end{aligned} \tag{B.2}$$

The LLG simulator represents a ferromagnetic thin film as a lattice of magnetic dipoles, each of which is spatially uniform inside a small unit cell. Hence, the kernel coefficients are also expressed as a discrete lattice. We use k and l as the spatial indices along the x and y directions respectively, and assume a single layer lattice in the z direction, centered at $z = 0$. In equation

B.2 above, the spatial coordinates x, y, z are then replaced by kd, ld, and 0 respectively. Defining the limits as

$$c_{x,i} = kd - \frac{d}{2} \qquad c_{y,i} = ld - \frac{d}{2} \qquad c_{z,i} = -\frac{h}{2}$$
$$c_{x,f} = kd + \frac{d}{2} \qquad c_{y,f} = ld + \frac{d}{2} \qquad c_{z,f} = +\frac{h}{2}$$

we finally get

$$\begin{aligned} G_{xx}(k,l) \;=\; & \arctan\left(\frac{c_{y,i}c_{z,i}}{c_{x,i}\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,i}^2}}\right) - \arctan\left(\frac{c_{y,i}c_{z,i}}{c_{x,f}\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,i}^2}}\right) \\ & - \arctan\left(\frac{c_{y,f}c_{z,i}}{c_{x,i}\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,i}^2}}\right) + \arctan\left(\frac{c_{y,f}c_{z,i}}{c_{x,f}\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,i}^2}}\right) \\ & - \arctan\left(\frac{c_{y,i}c_{z,f}}{c_{x,i}\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,f}^2}}\right) + \arctan\left(\frac{c_{y,i}c_{z,f}}{c_{x,f}\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,f}^2}}\right) \\ & + \arctan\left(\frac{c_{y,f}c_{z,f}}{c_{x,i}\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,f}^2}}\right) - \arctan\left(\frac{c_{y,f}c_{z,f}}{c_{x,f}\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,f}^2}}\right) \end{aligned} \tag{B.3}$$

Equation B.3 is the same formula that has been used to define $G_{xx}(k,l)$ in the LLG simulator source code (see Appendix C for the LLG simulator source code in C). Following the procedures outlined above, the rest of the coefficients can be determined in a similar manner. We list them below for completeness.

$$
\begin{aligned}
G_{yy}(k,l) \;=\; & \arctan\left(\frac{c_{x,i}c_{z,i}}{c_{y,i}\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,i}^2}}\right) - \arctan\left(\frac{c_{x,f}c_{z,i}}{c_{y,i}\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,i}^2}}\right) \\
& - \arctan\left(\frac{c_{x,i}c_{z,i}}{c_{y,f}\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,i}^2}}\right) + \arctan\left(\frac{c_{x,f}c_{z,i}}{c_{y,f}\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,i}^2}}\right) \\
& - \arctan\left(\frac{c_{x,i}c_{z,f}}{c_{y,i}\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,f}^2}}\right) + \arctan\left(\frac{c_{x,f}c_{z,f}}{c_{y,i}\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,f}^2}}\right) \\
& + \arctan\left(\frac{c_{x,i}c_{z,f}}{c_{y,f}\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,f}^2}}\right) - \arctan\left(\frac{c_{x,f}c_{z,f}}{c_{y,f}\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,f}^2}}\right)
\end{aligned}
\tag{B.4}
$$

$$
\begin{aligned}
G_{zz}(k,l) \;=\; & \arctan\left(\frac{c_{x,i}c_{y,i}}{c_{z,i}\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,i}^2}}\right) - \arctan\left(\frac{c_{x,f}c_{y,i}}{c_{z,i}\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,i}^2}}\right) \\
& - \arctan\left(\frac{c_{x,i}c_{y,f}}{c_{z,i}\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,i}^2}}\right) + \arctan\left(\frac{c_{x,f}c_{y,f}}{c_{z,i}\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,i}^2}}\right) \\
& - \arctan\left(\frac{c_{x,i}c_{y,i}}{c_{z,f}\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,f}^2}}\right) + \arctan\left(\frac{c_{x,f}c_{y,i}}{c_{z,f}\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,f}^2}}\right) \\
& + \arctan\left(\frac{c_{x,i}c_{y,f}}{c_{z,f}\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,f}^2}}\right) - \arctan\left(\frac{c_{x,f}c_{y,f}}{c_{z,f}\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,f}^2}}\right)
\end{aligned}
\tag{B.5}
$$

$$
\begin{aligned}
G_{xy}(k,l) \;=\; & G_{yx}(k,l) = -\ln\left(c_{z,i}+\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,i}^2}\right) + \ln\left(c_{z,i}+\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,i}^2}\right) \\
& + \ln\left(c_{z,i}+\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,i}^2}\right) - \ln\left(c_{z,i}+\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,i}^2}\right) \\
& + \ln\left(c_{z,f}+\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,f}^2}\right) - \ln\left(c_{z,f}+\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,f}^2}\right) \\
& - \ln\left(c_{z,f}+\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,f}^2}\right) + \ln\left(c_{z,f}+\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,f}^2}\right)
\end{aligned}
\tag{B.6}
$$

$$
\begin{aligned}
G_{xz}(k,l) &= G_{zx}(k,l) = -\ln\left(c_{y,i}+\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,i}^2}\right)+\ln\left(c_{y,i}+\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,i}^2}\right) \\
&+\ln\left(c_{y,f}+\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,i}^2}\right)-\ln\left(c_{y,f}+\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,i}^2}\right) \\
&+\ln\left(c_{y,i}+\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,f}^2}\right)-\ln\left(c_{y,i}+\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,f}^2}\right) \\
&-\ln\left(c_{y,f}+\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,f}^2}\right)+\ln\left(c_{y,f}+\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,f}^2}\right) \qquad \text{(B.7)}
\end{aligned}
$$

$$
\begin{aligned}
G_{yz}(k,l) &= G_{zy}(k,l) = -\ln\left(c_{x,i}+\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,i}^2}\right)+\ln\left(c_{x,f}+\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,i}^2}\right) \\
&+\ln\left(c_{x,i}+\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,i}^2}\right)-\ln\left(c_{x,f}+\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,i}^2}\right) \\
&+\ln\left(c_{x,i}+\sqrt{c_{x,i}^2+c_{y,i}^2+c_{z,f}^2}\right)-\ln\left(c_{x,f}+\sqrt{c_{x,f}^2+c_{y,i}^2+c_{z,f}^2}\right) \\
&-\ln\left(c_{x,i}+\sqrt{c_{x,i}^2+c_{y,f}^2+c_{z,f}^2}\right)+\ln\left(c_{x,f}+\sqrt{c_{x,f}^2+c_{y,f}^2+c_{z,f}^2}\right) \qquad \text{(B.8)}
\end{aligned}
$$

Bibliography

[1] Makhlouf Redjdal, *A Numerical Investigation of the Dynamics of Non-Periodic Micromagnetic Structures*, Ph.D Thesis, Boston University (1998)

Appendix C

Source Codes

In this Appendix, we provide a listing of the relevant computer software code that was used to numerically simulate the dynamics of magnetization in a ferromagnetic thin film. The files with the `.m` extension correspond mainly to MATLAB^{TM} programs that are used to create the graphical user interface (GUI) of the simulator. The main body of the simulation source code is listed in `llgincfft0928.c`.

```
# include <conio.h>
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
# include "F:/MSDEV/include/nrutil.h"
# include "F:/MSDEV/include/nrutil.c"
# include "F:/Matlab/extern/include/mex.h"
# include <math.h>
# define pi 3.14159265358979
# define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr
# define float double

double rows, columns, time_step, time_limit;
double alpha, gamma, d, M, K_u, A_x, Hx, Hy, Hz, width;
double dc, finish, displayfreq, toleranceX;
int displayfrequency;
unsigned int buflen=0;
char *buf;
double *foome_x, *foome_y, *foome_z;
double W_ext_at_time_t, W_ani_at_time_t, W_xch_at_time_t;


// addition:
double **M_X, **M_Y, **M_Z;
double time;
int MAXROWS, MAXCOLS;
double H_X, H_Y, H_Z;
double **H_total_X, **H_total_Y, **H_total_Z, **delta_M_X, **delta_M_Y, **delta_M_Z;

double H_dmag_X_total, H_dmag_Y_total, H_dmag_Z_total;

double fac, r_x, i_x, r_y, i_y, r_z, i_z, ***M_X_K, ***M_Y_K, ***M_Z_K, **spec_X, **spec
_Y, **spec_Z, *sp1_x, *sp2_x, *sp1_y, *sp2_y, *sp1_z, *sp2_z;
double *sp_Hx, *sp_Hy, *sp_Hz;
double ***G_00, ***G_01, ***G_02, ***G_10, ***G_11, ***G_12, ***G_20, ***G_21, ***G_22;
double **spec_G_00, **spec_G_01, **spec_G_02, **spec_G_10, **spec_G_11, **spec_G_12, **s
pec_G_20, **spec_G_21, **spec_G_22;
double ***H_X_K, ***H_Y_K, ***H_Z_K;
double **spec_H_X_K, **spec_H_Y_K, **spec_H_Z_K;
double H_ani_X=0.0, H_ani_Y=0.0, H_ani_Z=0.0, H_xch_X, H_xch_Y, H_xch_Z, H_xch_X_total,
H_xch_Y_total, H_xch_Z_total;
double temp_xch=0.0;
double temp_X, temp_Y, temp_Z, oldtempX;

double magnitude=1.0, aveX=0.0, aveY=0.0, aveZ=0.0, tempave=0.0;
double constant, h;
int myrows, mycolumns;
double timesofar=0.0, geriyekalan=0.0;
double replenishtime_limit;
double *meanmeanMX, *meanmeanMY, *meanmeanMZ, totalenergy=0.0, *totalenergy_pt;




//
void myllg(double *X_pt, double *Y_pt, double *Z_pt, double *points_X_pt, double *points
_Y_pt, double *points_Z_pt, double *W_ext_pt, double *W_ani_pt, double *W_xch_pt, double
 *modifiedtime, double *points_aveX_pt, double *points_aveY_pt, double *points_aveZ_pt)
{



        void rlft3(float ***data, float **spec, unsigned long nn1, unsigned long nn2, un
signed long nn3, int isign);
        void right_hand_side();

        int k, l, kk, ll;
        int countfoome=0, validcellcount=0, time_count=0, time_index=0, initcount=0;

        double z1i, z2i, z3i, z1f, z2f, z3f;
        double mtime_step;
        double **rightsideX_n_2, **rightsideY_n_2, **rightsideZ_n_2;
        double **rightsideX_n_1, **rightsideY_n_1, **rightsideZ_n_1;
        double **rightsideX_n_0, **rightsideY_n_0, **rightsideZ_n_0;
        double **rightsideX_n1, **rightsideY_n1, **rightsideZ_n1;
        double **M_X_n, **M_Y_n, **M_Z_n, **M_X_n_1, **M_Y_n_1, **M_Z_n_1;
        char mybuffer1[200], mybuffer2[15];
        int desiredkk=1, desiredll=1;
        int *myvalidcellcount;
        bool finecondition=false;

        FILE *fp;



# define dsquare d*d
# define M1 M


        time = 0.0;
        constant = gamma/(1+(alpha*alpha));
        myrows=(int)rows;
        mycolumns=(int)columns;
        mtime_step=time_step/200.0;


        MAXROWS=2*myrows;
        MAXCOLS=2*mycolumns;

        myvalidcellcount=ivector(1,1);



        M_X=dmatrix(1, myrows, 1, mycolumns);
        M_Y=dmatrix(1, myrows, 1, mycolumns);
        M_Z=dmatrix(1, myrows, 1, mycolumns);
        M_X_n=dmatrix(1, myrows, 1, mycolumns);
        M_Y_n=dmatrix(1, myrows, 1, mycolumns);
        M_Z_n=dmatrix(1, myrows, 1, mycolumns);
        M_X_n_1=dmatrix(1, myrows, 1, mycolumns);
        M_Y_n_1=dmatrix(1, myrows, 1, mycolumns);
        M_Z_n_1=dmatrix(1, myrows, 1, mycolumns);

        delta_M_X=dmatrix(1, myrows, 1, mycolumns);
        delta_M_Y=dmatrix(1, myrows, 1, mycolumns);
        delta_M_Z=dmatrix(1, myrows, 1, mycolumns);
        rightsideX_n_2=dmatrix(1, myrows, 1, mycolumns);
        rightsideY_n_2=dmatrix(1, myrows, 1, mycolumns);
        rightsideZ_n_2=dmatrix(1, myrows, 1, mycolumns);
        rightsideX_n_1=dmatrix(1, myrows, 1, mycolumns);
        rightsideY_n_1=dmatrix(1, myrows, 1, mycolumns);
        rightsideZ_n_1=dmatrix(1, myrows, 1, mycolumns);
        rightsideX_n_0=dmatrix(1, myrows, 1, mycolumns);
        rightsideY_n_0=dmatrix(1, myrows, 1, mycolumns);
        rightsideZ_n_0=dmatrix(1, myrows, 1, mycolumns);
        rightsideX_n1=dmatrix(1, myrows, 1, mycolumns);
        rightsideY_n1=dmatrix(1, myrows, 1, mycolumns);
        rightsideZ_n1=dmatrix(1, myrows, 1, mycolumns);
```

```
	printf("(int)rows: %i\n",myrows);
	//



	H_total_X=dmatrix(1,myrows,1,mycolumns);
	H_total_Y=dmatrix(1,myrows,1,mycolumns);
	H_total_Z=dmatrix(1,myrows,1,mycolumns);


	H_X_K=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	H_Y_K=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	H_Z_K=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);

	M_X_K=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	M_Y_K=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	M_Z_K=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	spec_X=dmatrix(1,1,1,2*MAXROWS);
	spec_Y=dmatrix(1,1,1,2*MAXROWS);
	spec_Z=dmatrix(1,1,1,2*MAXROWS);
	spec_H_X_K=dmatrix(1,1,1,2*MAXROWS);
	spec_H_Y_K=dmatrix(1,1,1,2*MAXROWS);
	spec_H_Z_K=dmatrix(1,1,1,2*MAXROWS);

	spec_G_00=dmatrix(1,1,1,2*MAXROWS);
	spec_G_01=dmatrix(1,1,1,2*MAXROWS);
	spec_G_02=dmatrix(1,1,1,2*MAXROWS);
	spec_G_10=dmatrix(1,1,1,2*MAXROWS);
	spec_G_11=dmatrix(1,1,1,2*MAXROWS);
	spec_G_12=dmatrix(1,1,1,2*MAXROWS);
	spec_G_20=dmatrix(1,1,1,2*MAXROWS);
	spec_G_21=dmatrix(1,1,1,2*MAXROWS);
	spec_G_22=dmatrix(1,1,1,2*MAXROWS);

	G_00=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	G_01=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	G_02=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	G_10=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	G_11=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	G_12=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	G_20=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	G_21=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);
	G_22=f3tensor(1,1,1,MAXROWS,1,MAXCOLS);

	printf("Memory allocation complete!\n");

	if (buflen > 3){
		fp=fopen(buf,"w");
	}

	countfoome=0;

	for (l=1;l<=mycolumns;l++){
		for (k=1;k<=myrows;k++){
			M_X[k][l] = *(foome_x+countfoome);
			M_Y[k][l] = *(foome_y+countfoome);
			M_Z[k][l] = *(foome_z+countfoome);
			countfoome++;
			if (!( (M_X[k][l]==0.0) && (M_Y[k][l]==0.0) && (M_Z[k][l]==0.0)
)){
				validcellcount++;
				desiredkk=k;
				desiredll=l;
```

```
			}

		}
	}

	printf("countfoome: %e, validcellcount: %e \n",countfoome, validcellcount);
	myvalidcellcount[1]=validcellcount;



	for (k=1;k<=MAXROWS; k++){
		for (l=1;l<=MAXCOLS; l++){

			G_00[1][k][l]=0.0;
			G_01[1][k][l]=0.0;
			G_02[1][k][l]=0.0;
			G_10[1][k][l]=0.0;
			G_11[1][k][l]=0.0;
			G_12[1][k][l]=0.0;
			G_20[1][k][l]=0.0;
			G_21[1][k][l]=0.0;
			G_22[1][k][l]=0.0;

			if ((k>=myrows+1) && (l>=mycolumns+1)){
				M_X_K[1][k][l]=M_X[k-myrows][l-mycolumns];
				M_Y_K[1][k][l]=M_Y[k-myrows][l-mycolumns];
				M_Z_K[1][k][l]=M_Z[k-myrows][l-mycolumns];
			}
			else{
				M_X_K[1][k][l]=0.0;
				M_Y_K[1][k][l]=0.0;
				M_Z_K[1][k][l]=0.0;
			}



		}
	}


	for (k=-myrows+1;k<=myrows-1; k++){
		for (l=-mycolumns+1;l<=mycolumns-1; l++){


			z1i= ((k)*d)-(d/2); ///((k*d)-(d/2));
			z2i= ((l)*d)-(d/2); ///((l*d)-(d/2));
			z3i= -h/2; ///(-d/2);

			z1f= ((k)*d)+(d/2);///((k*d)+(d/2));
			z2f= ((l)*d)+(d/2);///((l*d)+(d/2));
			z3f= h/2;///(d/2);

			//printf("%e  %e  %e  %e  %e  %e\n",z1i, z2i, z3i, z1f, z2f,
 z3f);

			k=k+myrows+1;
			l=l+mycolumns+1;


			G_00[1][k][l] =(float)(atan2(z2i*z3i , (z1i*pow((z1i*z1i)+(z
2i*z2i)+(z3i*z3i), 0.5)))
				- atan2(z2i*z3i ,( z1f*pow((z1f*z1f)+(z2i*z2i)+(z3i*
z3i), 0.5)))
				- atan2(z2f*z3i ,( z1i*pow((z1i*z1i)+(z2f*z2f)+(z3i*
```

```
z3i), 0.5)))
                        + atan2(z2f*z3i ,( z1f*pow((z1f*z1f)+(z2f*z2f)+(z3i*z3i)
, 0.5)))
                        - atan2(z2i*z3f ,( z1i*pow((z1i*z1i)+(z2i*z2i)+(z3f*z3f)
, 0.5)))
                        + atan2(z2i*z3f ,( z1f*pow((z1f*z1f)+(z2i*z2i)+(z3f*z3f)
, 0.5)))
                        + atan2(z2f*z3f ,( z1i*pow((z1i*z1i)+(z2f*z2f)+(z3f*z3f)
, 0.5)))
                        - atan2(z2f*z3f ,( z1f*pow((z1f*z1f)+(z2f*z2f)+(z3f*z3f)
, 0.5)))    );

                G_11[1][k][l] =(float)(atan2(z1i*z3i , (z2i*pow((z2i*z2i)+(z1i*z
1i)+(z3i*z3i), 0.5)))
                        - atan2(z1i*z3i ,( z2f*pow((z2f*z2f)+(z1i*z1i)+(z3i*z3i)
, 0.5)))
                        - atan2(z1f*z3i ,( z2i*pow((z2i*z2i)+(z1f*z1f)+(z3i*z3i)
, 0.5)))
                        + atan2(z1f*z3i ,( z2f*pow((z2f*z2f)+(z1f*z1f)+(z3i*z3i)
, 0.5)))
                        - atan2(z1i*z3f ,( z2i*pow((z2i*z2i)+(z1i*z1i)+(z3f*z3f)
, 0.5)))
                        + atan2(z1i*z3f ,( z2f*pow((z2f*z2f)+(z1i*z1i)+(z3f*z3f)
, 0.5)))
                        + atan2(z1f*z3f ,( z2i*pow((z2i*z2i)+(z1f*z1f)+(z3f*z3f)
, 0.5)))
                        - atan2(z1f*z3f ,( z2f*pow((z2f*z2f)+(z1f*z1f)+(z3f*z3f)
, 0.5)))    );

                G_22[1][k][l] =(float)(atan2(z1i*z2i , (z3i*pow((z3i*z3i)+(z1i*z
1i)+(z2i*z2i), 0.5)))
                        - atan2(z1i*z2i ,( z3f*pow((z3f*z3f)+(z1i*z1i)+(z2i*z2i)
, 0.5)))
                        - atan2(z1f*z2i ,( z3i*pow((z3i*z3i)+(z1f*z1f)+(z2i*z2i)
, 0.5)))
                        + atan2(z1f*z2i ,( z3f*pow((z3f*z3f)+(z1f*z1f)+(z2i*z2i)
, 0.5)))
                        - atan2(z1i*z2f ,( z3i*pow((z3i*z3i)+(z1i*z1i)+(z2f*z2f)
, 0.5)))
                        + atan2(z1i*z2f ,( z3f*pow((z3f*z3f)+(z1i*z1i)+(z2f*z2f)
, 0.5)))
                        + atan2(z1f*z2f ,( z3i*pow((z3i*z3i)+(z1f*z1f)+(z2f*z2f)
, 0.5)))
                        - atan2(z1f*z2f ,( z3f*pow((z3f*z3f)+(z1f*z1f)+(z2f*z2f)
, 0.5)))    );


                G_01[1][k][l] =(float)(-log(z3i+pow((z1i*z1i)+(z2i*z2i)+(z3i*z3i
), 0.5))
                        + log(z3i+pow((z1f*z1f)+(z2i*z2i)+(z3i*z3i), 0.5))
                        + log(z3i+pow((z1i*z1i)+(z2f*z2f)+(z3i*z3i), 0.5))
                        - log(z3i+pow((z1f*z1f)+(z2f*z2f)+(z3i*z3i), 0.5))
                        + log(z3f+pow((z1i*z1i)+(z2i*z2i)+(z3f*z3f), 0.5))
                        - log(z3f+pow((z1f*z1f)+(z2i*z2i)+(z3f*z3f), 0.5))
                        - log(z3f+pow((z1i*z1i)+(z2f*z2f)+(z3f*z3f), 0.5))
                        + log(z3f+pow((z1f*z1f)+(z2f*z2f)+(z3f*z3f), 0.5))  );

                G_10[1][k][l] =G_01[1][k][l];


                G_02[1][k][l] =(float)(-log(z2i+pow((z1i*z1i)+(z3i*z3i)+(z2i*z2i
), 0.5))
                        + log(z2i+pow((z1f*z1f)+(z3i*z3i)+(z2i*z2i), 0.5))
                        + log(z2i+pow((z1i*z1i)+(z3f*z3f)+(z2i*z2i), 0.5))
                        - log(z2i+pow((z1f*z1f)+(z3f*z3f)+(z2i*z2i), 0.5))
                        + log(z2f+pow((z1i*z1i)+(z3i*z3i)+(z2f*z2f), 0.5))
                        - log(z2f+pow((z1f*z1f)+(z3i*z3i)+(z2f*z2f), 0.5))
                        - log(z2f+pow((z1i*z1i)+(z3f*z3f)+(z2f*z2f), 0.5))
                        + log(z2f+pow((z1f*z1f)+(z3f*z3f)+(z2f*z2f), 0.5))
);


                G_20[1][k][l] =G_02[1][k][l];

                G_12[1][k][l] =(float)(-log(z1i+pow((z3i*z3i)+(z2i*z2i)+(z1i
*z1i), 0.5))
                        + log(z1i+pow((z3f*z3f)+(z2i*z2i)+(z1i*z1i), 0.5))
                        + log(z1i+pow((z3i*z3i)+(z2f*z2f)+(z1i*z1i), 0.5))
                        - log(z1i+pow((z3f*z3f)+(z2f*z2f)+(z1i*z1i), 0.5))
                        + log(z1f+pow((z3i*z3i)+(z2i*z2i)+(z1f*z1f), 0.5))
                        - log(z1f+pow((z3f*z3f)+(z2i*z2i)+(z1f*z1f), 0.5))
                        - log(z1f+pow((z3i*z3i)+(z2f*z2f)+(z1f*z1f), 0.5))
                        + log(z1f+pow((z3f*z3f)+(z2f*z2f)+(z1f*z1f), 0.5))
);


                G_21[1][k][l] =G_12[1][k][l];

                k=k-myrows-1;
                l=l-mycolumns-1;

            }
        }

        printf("Initial Factors G_00: %e  %e  %e  %e\n",G_00[1][1][1], G_00[1][1][2]
, G_00[1][2][1], G_00[1][2][2]);
        printf("Initial Factors G_11: %e  %e  %e  %e\n",G_11[1][1][1], G_11[1][1][2]
, G_11[1][2][1], G_11[1][2][2]);
        printf("Initial Factors G_22: %e  %e  %e  %e\n",G_22[1][1][1], G_22[1][1][2]
, G_22[1][2][1], G_22[1][2][2]);
        printf("Initial Factors G_01: %e  %e  %e  %e\n",G_01[1][1][1], G_01[1][1][2]
, G_01[1][2][1], G_01[1][2][2]);
        printf("Initial Factors G_02: %e  %e  %e  %e\n",G_02[1][1][1], G_02[1][1][2]
, G_02[1][2][1], G_02[1][2][2]);
        printf("Initial Factors G_10: %e  %e  %e  %e\n",G_10[1][1][1], G_10[1][1][2]
, G_10[1][2][1], G_10[1][2][2]);
        printf("Initial Factors G_12: %e  %e  %e  %e\n",G_12[1][1][1], G_12[1][1][2]
, G_12[1][2][1], G_12[1][2][2]);
        printf("Initial Factors G_21: %e  %e  %e  %e\n",G_21[1][1][1], G_21[1][1][2]
, G_21[1][2][1], G_21[1][2][2]);
        printf("Initial Factors G_20: %e  %e  %e  %e\n",G_20[1][1][1], G_20[1][1][2]
, G_20[1][2][1], G_20[1][2][2]);

        printf("Starting FFT Initialization!\n");
        rlft3(G_00, spec_G_00,1,MAXROWS,MAXCOLS,1);
        printf("First FFT Initialization Complete!\n");
        rlft3(G_01, spec_G_01,1,MAXROWS,MAXCOLS,1);
        printf("Second FFT Initialization Complete!\n");
        rlft3(G_02, spec_G_02,1,MAXROWS,MAXCOLS,1);
        printf("Third FFT Initialization Complete!\n");
        rlft3(G_10, spec_G_10,1,MAXROWS,MAXCOLS,1);
        printf("Fourth FFT Initialization Complete!\n");
        rlft3(G_11, spec_G_11,1,MAXROWS,MAXCOLS,1);
        printf("Fifth FFT Initialization Complete!\n");
        rlft3(G_12, spec_G_12,1,MAXROWS,MAXCOLS,1);
        printf("Sixth FFT Initialization Complete!\n");
        rlft3(G_20, spec_G_20,1,MAXROWS,MAXCOLS,1);
        printf("Seventh FFT Initialization Complete!\n");
        rlft3(G_21, spec_G_21,1,MAXROWS,MAXCOLS,1);
```

```
        printf("Eighth FFT Initialization Complete!\n");
        rlft3(G_22, spec_G_22,1,MAXROWS,MAXCOLS,1);
        printf("Nineth FFT Initialization Complete!\n");

        fac=2.0/(1.0*MAXROWS*MAXCOLS);
        printf("Initializing FFT's complete\n");
        printf("FFTed Factors: %e %e %e %e %e %e %e %e %e %e %e\n",G_00[1][1][
1], G_00[1][2][1], G_00[1][1][2], G_01[1][1][1], G_02[1][1][1], G_10[1][1][1], G_11[1][1
][1], G_12[1][1][1], G_20[1][1][1], G_21[1][1][1], G_22[1][1][1]);

        time=0.0;

        while (1){
                // inside the while loop:
                timesofar=time;

                geriyekalan=min(replenishtime_limit, time_limit-time);
                if (geriyekalan < 0){
                        break;}

                for (time=timesofar;time<=2.0*time_step+mtime_step+timesofar;time=time+m
time_step)
                {

                        time_count++;
                        //time_index++;
                        right_hand_side();

                        aveX=0.0;
                        aveY=0.0;
                        aveZ=0.0;

                        if (time==timesofar){
                                for (kk=1;kk<=myrows;kk++){
                                        for (ll=1;ll<=mycolumns;ll++){
                                                delta_M_X[kk][ll] = mtime_step*constant*
( ((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll] - M_Y[kk][ll]*(H_total_Z[kk][ll]
+(M*H_Z_K[1][kk][ll]))) + alpha*( (M_Y[kk][ll]*((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))
*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll])))) - (M_Z[kk][ll]*((H_t
otal_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K
[1][kk][ll])))) ) );
                                                delta_M_Y[kk][ll] = mtime_step*constant*
( ((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll] - M_Z[kk][ll]*(H_total_X[kk][ll]
+(M*H_X_K[1][kk][ll]))) + alpha*( (M_Z[kk][ll]*((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))
*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll])))) - (M_X[kk][ll]*((H_t
otal_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K
[1][kk][ll])))) ) );
                                                delta_M_Z[kk][ll] = mtime_step*constant*
( ((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll] - M_X[kk][ll]*(H_total_Y[kk][ll]
+(M*H_Y_K[1][kk][ll]))) + alpha*( (M_X[kk][ll]*((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))
*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk][ll])))) - (M_Y[kk][ll]*((H_t
otal_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K
[1][kk][ll])))) ) );
                                                //M_X[kk][ll] = M_X[kk][ll] + delta_M_X[
kk][ll];
                                                //M_Y[kk][ll] = M_Y[kk][ll] + delta_M_Y[
kk][ll];
                                                //M_Z[kk][ll] = M_Z[kk][ll] + delta_M_Z[
kk][ll];


                                                aveX=aveX+M_X[kk][ll];
                                                aveY=aveY+M_Y[kk][ll];
                                                aveZ=aveZ+M_Z[kk][ll];
                                                rightsideX_n_2[kk][ll] = constant*(
((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll] - M_Y[kk][ll]*(H_total_Z[kk][l
l]+(M*H_Z_K[1][kk][ll]))) + alpha*( (M_Y[kk][ll]*((H_total_X[kk][ll]+(M*H_X_K[1][kk]
[ll]))*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll])))) - (M_Z[kk]
[ll]*((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk
][ll]+(M*H_X_K[1][kk][ll])))) ) );
                                                rightsideY_n_2[kk][ll] = constant*(
((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll] - M_Z[kk][ll]*(H_total_X[kk][l
l]+(M*H_X_K[1][kk][ll]))) + alpha*( (M_Z[kk][ll]*((H_total_Y[kk][ll]+(M*H_Y_K[1][kk]
[ll]))*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll])))) - (M_X[kk]
[ll]*((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk
][ll]+(M*H_Y_K[1][kk][ll])))) ) );
                                                rightsideZ_n_2[kk][ll] = constant*(
((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll] - M_X[kk][ll]*(H_total_Y[kk][l
l]+(M*H_Y_K[1][kk][ll]))) + alpha*( (M_X[kk][ll]*((H_total_Z[kk][ll]+(M*H_Z_K[1][kk]
[ll]))*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk][ll])))) - (M_Y[kk]
[ll]*((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk
][ll]+(M*H_Z_K[1][kk][ll])))) ) );

                                        }
                                }
                        }
                        else{
                                for (kk=1;kk<=myrows;kk++){
                                        for (ll=1;ll<=mycolumns;ll++){
                                                temp_X = mtime_step*constant*( ((H_t
otal_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll] - M_Y[kk][ll]*(H_total_Z[kk][ll]+(M
*H_Z_K[1][kk][ll]))) + alpha*( (M_Y[kk][ll]*((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll])
)*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll])))) - (M_Z[kk][ll]*
((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]
+(M*H_X_K[1][kk][ll])))) ) );
                                                temp_Y = mtime_step*constant*( ((H_t
otal_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll] - M_Z[kk][ll]*(H_total_X[kk][ll]+(M
*H_X_K[1][kk][ll]))) + alpha*( (M_Z[kk][ll]*((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll])
)*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll])))) - (M_X[kk][ll]*
((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]
+(M*H_Y_K[1][kk][ll])))) ) );
                                                temp_Z = mtime_step*constant*( ((H_t
otal_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll] - M_X[kk][ll]*(H_total_Y[kk][ll]+(M
*H_Y_K[1][kk][ll]))) + alpha*( (M_X[kk][ll]*((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll])
)*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk][ll])))) - (M_Y[kk][ll]*
((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]
+(M*H_Z_K[1][kk][ll])))) ) );


                                                delta_M_X[kk][ll] = 0.5*(delta_M_X[k
k][ll] + temp_X);
                                                delta_M_Y[kk][ll] = 0.5*(delta_M_Y[k
k][ll] + temp_Y);
                                                delta_M_Z[kk][ll] = 0.5*(delta_M_Z[k
k][ll] + temp_Z);


                                                M_X[kk][ll] = M_X[kk][ll] + delta_M_
X[kk][ll];
                                                M_Y[kk][ll] = M_Y[kk][ll] + delta_M_
Y[kk][ll];
                                                M_Z[kk][ll] = M_Z[kk][ll] + delta_M_
Z[kk][ll];


                                                delta_M_X[kk][ll] = temp_X;
                                                delta_M_Y[kk][ll] = temp_Y;
                                                delta_M_Z[kk][ll] = temp_Z;


                                                if (!( (M_X[kk][ll]==0.0) && (M_Y[kk
```

```
][ll]==0.0) && (M_Z[kk][ll]==0.0) ){
                                                magnitude = pow((1/(M_X[kk][ll]*
M_X[kk][ll] + M_Y[kk][ll]*M_Y[kk][ll] +M_Z[kk][ll]*M_Z[kk][ll])),0.5);
                                        }
                                        M_X[kk][ll]=magnitude*M_X[kk][ll];
                                        M_Y[kk][ll]=magnitude*M_Y[kk][ll];
                                        M_Z[kk][ll]=magnitude*M_Z[kk][ll];

                                        aveX=aveX+M_X[kk][ll];
                                        aveY=aveY+M_Y[kk][ll];
                                        aveZ=aveZ+M_Z[kk][ll];

                                        // ********
                                        if (time_step-mtime_step+timesofar<=time
<=time_step+timesofar){
                                                rightsideX_n_1[kk][ll] = constan
t*( ((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll] - M_Y[kk][ll]*(H_total_Z[kk][l
l]+(M*H_Z_K[1][kk][ll])))) + alpha*( (M_Y[kk][ll]*((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]
))*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll])))) - (M_Z[kk][ll]*((H
_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X
_K[1][kk][ll])))) ) );
                                                rightsideY_n_1[kk][ll] = constan
t*( ((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll] - M_Z[kk][ll]*(H_total_X[kk][l
l]+(M*H_X_K[1][kk][ll])))) + alpha*( (M_Z[kk][ll]*((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]
))*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll])))) - (M_X[kk][ll]*((H
_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y
_K[1][kk][ll])))) ) );
                                                rightsideZ_n_1[kk][ll] = constan
t*( ((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll] - M_X[kk][ll]*(H_total_Y[kk][l
l]+(M*H_Y_K[1][kk][ll])))) + alpha*( (M_X[kk][ll]*((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]
))*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk][ll])))) - (M_Y[kk][ll]*((H
_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z
_K[1][kk][ll])))) ) );
                                                M_X_n_1[kk][ll]=M_X[kk][ll];
                                                M_Y_n_1[kk][ll]=M_Y[kk][ll];
                                                M_Z_n_1[kk][ll]=M_Z[kk][ll];


                                        }

                                        if (2*time_step-mtime_step+timesofar<=ti
me<=2*time_step+timesofar){
                                                rightsideX_n_0[kk][ll] = constan
t*( ((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll] - M_Y[kk][ll]*(H_total_Z[kk][l
l]+(M*H_Z_K[1][kk][ll])))) + alpha*( (M_Y[kk][ll]*((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]
))*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll])))) - (M_Z[kk][ll]*((H
_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X
_K[1][kk][ll])))) ) );
                                                rightsideY_n_0[kk][ll] = constan
t*( ((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll] - M_Z[kk][ll]*(H_total_X[kk][l
l]+(M*H_X_K[1][kk][ll])))) + alpha*( (M_Z[kk][ll]*((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]
))*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll])))) - (M_X[kk][ll]*((H
_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y
_K[1][kk][ll])))) ) );
                                                rightsideZ_n_0[kk][ll] = constan
t*( ((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll] - M_X[kk][ll]*(H_total_Y[kk][l
l]+(M*H_Y_K[1][kk][ll])))) + alpha*( (M_X[kk][ll]*((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]
))*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk][ll])))) - (M_Y[kk][ll]*((H
_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z
_K[1][kk][ll])))) ) );

                                                M_X_n[kk][ll]=M_X[kk][ll];
                                                M_Y_n[kk][ll]=M_Y[kk][ll];
                                                M_Z_n[kk][ll]=M_Z[kk][ll];


                                        }
                                                                        // ********
                                                }
                                        }
                                }


                                tempave=aveX;
                                aveX=tempave/myvalidcellcount[1];
                                tempave=aveY;
                                aveY=tempave/myvalidcellcount[1];
                                tempave=aveZ;
                                aveZ=tempave/myvalidcellcount[1];



                                // calculate the energy due to the interaction of M with H_e
xt:


                                //fprintf(fp,"%5e \n", W_ext[time_index]);
                                /* *(W_ext_pt+time_index-1)=W_ext_at_time_t;
                                *(W_ani_pt+time_index-1)=W_ani_at_time_t;
                                *(W_xch_pt+time_index-1)=W_xch_at_time_t;
                                *(points_X_pt+time_index-1)=M_X[k-1][1-1];
                                *(points_Y_pt+time_index-1)=M_Y[k-1][1-1];
                                *(points_Z_pt+time_index-1)=M_Z[k-1][1-1];
                                *(points_aveX_pt+time_index-1)=aveX;
                                *(points_aveY_pt+time_index-1)=aveY;
                                *(points_aveZ_pt+time_index-1)=aveZ;


                                        *(modifiedtime+time_index-1)=time;
                                */
                                if (time_count==displayfrequency){
                                        time_count=0;
                                        //printf("%e\t %e\t %e\t %e\t %e\n", time, W_ext_at_
time_t, H_X_K[1][1][1], H_Y_K[1][1][1], H_Z_K[1][1][1]);
                                        _gcvt( time, 7, mybuffer2 );
                                        strcpy(mybuffer1,"set(findobj(IntegratorHandle,'Tag'
,'EditText19'),'String',");
                                        strcat(mybuffer1,mybuffer2);
                                        strcat(mybuffer1," ); drawnow;");
                                        mexEvalString(mybuffer1);
                                        _gcvt( aveX, 9, mybuffer2 );
                                        strcpy(mybuffer1,"set(findobj(IntegratorHandle,'Tag'
,'EditText20'),'String',");
                                        strcat(mybuffer1,mybuffer2);
                                        strcat(mybuffer1," ); drawnow;");
                                        mexEvalString(mybuffer1);

                                        /*if (buflen > 3){
                                        for (k=1;k<=myrows;k++){
                                        for (l=1;l<=mycolumns;l++){
                                        fprintf(fp, "%e  %e  %e  %e\n", time, M_X[k][l], M_Y
[k][l], M_Z[k][l]);
                                        }
                                        }
                                }*/

                                }

        }
```

```
        // now start the predictor-corrector steps:

        geriyekalan=min(replenishtime_limit, time_limit-time);
        if (geriyekalan < 0){
                break;}

        for (time=3*time_step+timesofar;time<=geriyekalan+timesofar;time=time+time_step)
{
                time_count++;
                time_index++;

                aveX=0.0;
                aveY=0.0;
                aveZ=0.0;

                for (kk=1;kk<=myrows;kk++){
                        for (ll=1;ll<=mycolumns;ll++){

                                // prediction:
                                M_X[kk][ll] = M_X_n[kk][ll] + ( (4.0*time_step/3.0)*(2.0
*rightsideX_n_0[kk][ll] - rightsideX_n_1[kk][ll] + 2.0*rightsideX_n_2[kk][ll] ) );
                                M_Y[kk][ll] = M_Y_n[kk][ll] + ( (4.0*time_step/3.0)*(2.0
*rightsideY_n_0[kk][ll] - rightsideY_n_1[kk][ll] + 2.0*rightsideY_n_2[kk][ll] ) );
                                M_Z[kk][ll] = M_Z_n[kk][ll] + ( (4.0*time_step/3.0)*(2.0
*rightsideZ_n_0[kk][ll] - rightsideZ_n_1[kk][ll] + 2.0*rightsideZ_n_2[kk][ll] ) );

                                if (!( (M_X[kk][ll]==0.0) && (M_Y[kk][ll]==0.0) && (M_Z[
kk][ll]==0.0) )){
                                        magnitude = pow((1/(M_X[kk][ll]*M_X[kk][ll] + M_
Y[kk][ll]*M_Y[kk][ll] +M_Z[kk][ll]*M_Z[kk][ll])),0.5);
                                }
                                M_X[kk][ll]=magnitude*M_X[kk][ll];
                                M_Y[kk][ll]=magnitude*M_Y[kk][ll];
                                M_Z[kk][ll]=magnitude*M_Z[kk][ll];

                                /*if (fabs(M_X[kk][ll]) > oldtempX){
                                oldtempX = fabs(M_X[kk][ll]);}*/

                        }
                }

                finecondition=false;
                while (!(finecondition)){// just out (fabs(newtempX-oldtempX) > toleranc
eX){
                        right_hand_side();
                        finecondition=true;
                        for (kk=1;kk<=myrows;kk++){
                                for (ll=1;ll<=mycolumns;ll++){

                                        // just in !!!!!!!!!!!!!!!!
                                        oldtempX=M_X[kk][ll];



                                        rightsideX_n1[kk][ll] = constant*( ((H_total_Y[k
k][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll] - M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K[1][kk
][ll]))) + alpha*( (M_Y[kk][ll]*((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll]-M_
X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll])))) - (M_Z[kk][ll]*((H_total_Z[kk][ll]+
(M*H_Z_K[1][kk][ll]))*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk][ll])))))
 ) );
                                        rightsideY_n1[kk][ll] = constant*( ((H_total_Z[k
k][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll] - M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk
][ll]))) + alpha*( (M_Z[kk][ll]*((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll]-M_
Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll])))) - (M_X[kk][ll]*((H_total_X[kk][ll]+
(M*H_X_K[1][kk][ll]))*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll])))))
 ) );
                                        rightsideZ_n1[kk][ll] = constant*( ((H_total_X[k
k][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll] - M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1][kk
][ll]))) + alpha*( (M_X[kk][ll]*((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll]-M_
Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk][ll])))) - (M_Y[kk][ll]*((H_total_Y[
 ) );
                                        rightsideZ_n1[kk][ll] = constant*( ((H_total_X[kk][l
_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll] - M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y
_K[1][kk][ll]))) + alpha*( (M_X[kk][ll]*((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_
X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk][ll])))) - (M_Y[kk][ll]*((H_
total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*
H_Z_K[1][kk][ll])))) ) );

                                        //correction:
                                        M_X[kk][ll] = M_X_n_1[kk][ll] + ( (time_step
/3.0)*(rightsideX_n1[kk][ll] + 4.0*rightsideX_n_0[kk][ll] + rightsideX_n_1[kk][ll])
);
                                        M_Y[kk][ll] = M_Y_n_1[kk][ll] + ( (time_step
/3.0)*(rightsideY_n1[kk][ll] + 4.0*rightsideY_n_0[kk][ll] + rightsideY_n_1[kk][ll])
);
                                        M_Z[kk][ll] = M_Z_n_1[kk][ll] + ( (time_step
/3.0)*(rightsideZ_n1[kk][ll] + 4.0*rightsideZ_n_0[kk][ll] + rightsideZ_n_1[kk][ll])
);

                                        //this about removing this normalization sec
tion:
                                        if (!( (M_X[kk][ll]==0.0) && (M_Y[kk][ll]==0
.0) && (M_Z[kk][ll]==0.0) )){
                                                magnitude = pow((1/(M_X[kk][ll]*M_X[
kk][ll] + M_Y[kk][ll]*M_Y[kk][ll] +M_Z[kk][ll]*M_Z[kk][ll])),0.5);
                                        }
                                        M_X[kk][ll]=magnitude*M_X[kk][ll];
                                        M_Y[kk][ll]=magnitude*M_Y[kk][ll];
                                        M_Z[kk][ll]=magnitude*M_Z[kk][ll];
                                        //*******


                                        /*if (fabs(M_X[kk][ll]) > newtempX){
                                        newtempX = fabs(M_X[kk][ll]);}*/
                                        //      printf("newtempX: %e\n",newtempX);

                                        if (fabs(M_X[kk][ll]-oldtempX) > toleranceX)
{
                                                finecondition=false;}

                                }
                        }
                }

                right_hand_side();
                totalenergy=0.0;

                for (kk=1;kk<=myrows;kk++){
                        for (ll=1;ll<=mycolumns;ll++){
                                rightsideX_n1[kk][ll] = constant*( ((H_total_Y[kk][l
l]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll] - M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K[1][kk
][ll]))) + alpha*( (M_Y[kk][ll]*((H_total_X[kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll
]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll])))) - (M_Z[kk][ll]*((H_total_Z[
kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1]
[kk][ll])))) ) );
                                rightsideY_n1[kk][ll] = constant*( ((H_total_Z[kk][l
l]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll] - M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk
][ll]))) + alpha*( (M_Z[kk][ll]*((H_total_Y[kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll
]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll])))) - (M_X[kk][ll]*((H_total_X[
kk][ll]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll]-M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1]
[kk][ll])))) ) );
                                rightsideZ_n1[kk][ll] = constant*( ((H_total_X[kk][l
l]+(M*H_X_K[1][kk][ll]))*M_Y[kk][ll] - M_X[kk][ll]*(H_total_Y[kk][ll]+(M*H_Y_K[1][kk
][ll]))) + alpha*( (M_X[kk][ll]*((H_total_Z[kk][ll]+(M*H_Z_K[1][kk][ll]))*M_X[kk][ll
]-M_Z[kk][ll]*(H_total_X[kk][ll]+(M*H_X_K[1][kk][ll])))) - (M_Y[kk][ll]*((H_total_Y[
```

```
kk][ll]+(M*H_Y_K[1][kk][ll]))*M_Z[kk][ll]-M_Y[kk][ll]*(H_total_Z[kk][ll]+(M*H_Z_K[1][kk]
[ll]))))  ) );

                                rightsideX_n_2[kk][ll]=rightsideX_n_1[kk][ll];
                                rightsideX_n_1[kk][ll]=rightsideX_n_0[kk][ll];
                                rightsideX_n_0[kk][ll]=rightsideX_n1[kk][ll];

                                rightsideY_n_2[kk][ll]=rightsideY_n_1[kk][ll];
                                rightsideY_n_1[kk][ll]=rightsideY_n_0[kk][ll];
                                rightsideY_n_0[kk][ll]=rightsideY_n1[kk][ll];

                                rightsideZ_n_2[kk][ll]=rightsideZ_n_1[kk][ll];
                                rightsideZ_n_1[kk][ll]=rightsideZ_n_0[kk][ll];
                                rightsideZ_n_0[kk][ll]=rightsideZ_n1[kk][ll];

                                M_X_n_1[kk][ll]=M_X_n[kk][ll];
                                M_Y_n_1[kk][ll]=M_Y_n[kk][ll];
                                M_Z_n_1[kk][ll]=M_Z_n[kk][ll];

                                M_X_n[kk][ll]=M_X[kk][ll];
                                M_Y_n[kk][ll]=M_Y[kk][ll];
                                M_Z_n[kk][ll]=M_Z[kk][ll];

                                aveX=aveX+M_X[kk][ll];
                                aveY=aveY+M_Y[kk][ll];
                                aveZ=aveZ+M_Z[kk][ll];

                                totalenergy=totalenergy-M*( (M_X[kk][ll]*H_total_X[kk][l
l])+(M_Y[kk][ll]*H_total_Y[kk][ll])+(M_Z[kk][ll]*H_total_Z[kk][ll]) );

                        }
                }



                tempave=aveX;
                aveX=tempave/myvalidcellcount[1];
                tempave=aveY;
                aveY=tempave/myvalidcellcount[1];
                tempave=aveZ;
                aveZ=tempave/myvalidcellcount[1];



                // calculate the energy due to the interaction of M with H_ext:


                //fprintf(fp,"%5e \n", W_ext[time_index]);
                *(W_ext_pt+time_index-1)=W_ext_at_time_t;
                *(W_ani_pt+time_index-1)=W_ani_at_time_t;
                *(W_xch_pt+time_index-1)=W_xch_at_time_t;
                *(points_X_pt+time_index-1)=M_X[k-1][l-1];
                *(points_Y_pt+time_index-1)=M_Y[k-1][l-1];
                *(points_Z_pt+time_index-1)=M_Z[k-1][l-1];
                *(points_aveX_pt+time_index-1)=aveX;
                *(points_aveY_pt+time_index-1)=aveY;
                *(points_aveZ_pt+time_index-1)=aveZ;
                *(totalenergy_pt+time_index-1)=totalenergy;


                *(modifiedtime+time_index-1)=time;

                if (time_count==displayfrequency){
                        time_count=0;
                        //printf("%e\t %e\t %e\t %e\t %e\n", time, W_ext_at_time_t, H_X_
```

```
K[1][1][1], H_Y_K[1][1][1], H_Z_K[1][1][1]);
                        _gcvt( time, 7, mybuffer2 );
                        strcpy(mybuffer1,"set(findobj(IntegratorHandle,'Tag','EditTe
xt19'),'String',");
                        strcat(mybuffer1,mybuffer2);
                        strcat(mybuffer1," ); drawnow;");
                        mexEvalString(mybuffer1);
                        _gcvt( aveX, 9, mybuffer2 );
                        strcpy(mybuffer1,"set(findobj(IntegratorHandle,'Tag','EditTe
xt20'),'String',");
                        strcat(mybuffer1,mybuffer2);
                        strcat(mybuffer1," ); drawnow;");
                        mexEvalString(mybuffer1);

                        if (buflen > 3){
                                for (k=1;k<=myrows;k++){
                                        for (l=1;l<=mycolumns;l++){
                                                fprintf(fp, "%e  %e  %e  %e\n", time
, M_X[k][l], M_Y[k][l], M_Z[k][l]);
                                        }
                                }
                        }

                }
}

}



// ***** this section must be at the end, outside the for loops

*(meanmeanMX)=aveX;
*(meanmeanMY)=aveY;
*(meanmeanMZ)=aveZ;


if (buflen > 3){
        fclose(fp);
}
initcount=0;
for (ll=1;ll<=mycolumns;ll++){
        for (kk=1;kk<=myrows;kk++){
                *(X_pt+initcount)=M_X[kk][ll];
                *(Y_pt+initcount)=M_Y[kk][ll];
                *(Z_pt+initcount)=M_Z[kk][ll];
                initcount++;
        }
}

free_dmatrix(H_total_X,1,myrows, 1, mycolumns);
free_dmatrix(H_total_Y,1,myrows, 1, mycolumns);
free_dmatrix(H_total_Z,1,myrows, 1, mycolumns);

free_dmatrix(M_X,1,myrows, 1, mycolumns);
free_dmatrix(M_Y,1,myrows, 1, mycolumns);
free_dmatrix(M_Z,1,myrows, 1, mycolumns);
free_dmatrix(M_X_n,1,myrows, 1, mycolumns);
free_dmatrix(M_Y_n,1,myrows, 1, mycolumns);
free_dmatrix(M_Z_n,1,myrows, 1, mycolumns);
free_dmatrix(M_X_n_1,1,myrows, 1, mycolumns);
free_dmatrix(M_Y_n_1,1,myrows, 1, mycolumns);
free_dmatrix(M_Z_n_1,1,myrows, 1, mycolumns);
```

```
free_dmatrix(delta_M_X,1,myrows, 1, mycolumns);
free_dmatrix(delta_M_Y,1,myrows, 1, mycolumns);
free_dmatrix(delta_M_Z,1,myrows, 1, mycolumns);

free_dmatrix(rightsideX_n_2,1,myrows, 1, mycolumns);
free_dmatrix(rightsideY_n_2,1,myrows, 1, mycolumns);
free_dmatrix(rightsideZ_n_2,1,myrows, 1, mycolumns);
free_dmatrix(rightsideX_n_1,1,myrows, 1, mycolumns);
free_dmatrix(rightsideY_n_1,1,myrows, 1, mycolumns);
free_dmatrix(rightsideZ_n_1,1,myrows, 1, mycolumns);
free_dmatrix(rightsideX_n_0,1,myrows, 1, mycolumns);
free_dmatrix(rightsideY_n_0,1,myrows, 1, mycolumns);
free_dmatrix(rightsideZ_n_0,1,myrows, 1, mycolumns);
free_dmatrix(rightsideX_n1,1,myrows, 1, mycolumns);
free_dmatrix(rightsideY_n1,1,myrows, 1, mycolumns);
free_dmatrix(rightsideZ_n1,1,myrows, 1, mycolumns);



free_f3tensor(G_22,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(G_21,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(G_20,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(G_12,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(G_11,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(G_10,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(G_02,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(G_01,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(G_00,1,1,1,MAXROWS,1,MAXCOLS);
free_dmatrix(spec_G_22,1,1,1,2*MAXROWS);
free_dmatrix(spec_G_21,1,1,1,2*MAXROWS);
free_dmatrix(spec_G_20,1,1,1,2*MAXROWS);
free_dmatrix(spec_G_12,1,1,1,2*MAXROWS);
free_dmatrix(spec_G_11,1,1,1,2*MAXROWS);
free_dmatrix(spec_G_10,1,1,1,2*MAXROWS);
free_dmatrix(spec_G_02,1,1,1,2*MAXROWS);
free_dmatrix(spec_G_01,1,1,1,2*MAXROWS);
free_dmatrix(spec_G_00,1,1,1,2*MAXROWS);

free_dmatrix(spec_Z,1,1,1,2*MAXROWS);
free_dmatrix(spec_Y,1,1,1,2*MAXROWS);
free_dmatrix(spec_X,1,1,1,2*MAXROWS);

free_dmatrix(spec_H_Z_K,1,1,1,2*MAXROWS);
free_dmatrix(spec_H_Y_K,1,1,1,2*MAXROWS);
free_dmatrix(spec_H_X_K,1,1,1,2*MAXROWS);

free_f3tensor(M_Z_K,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(M_Y_K,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(M_X_K,1,1,1,MAXROWS,1,MAXCOLS);

free_f3tensor(H_Z_K,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(H_Y_K,1,1,1,MAXROWS,1,MAXCOLS);
free_f3tensor(H_X_K,1,1,1,MAXROWS,1,MAXCOLS);

free_ivector(myvalidcellcount,1,1);


printf("All matrices freed!\n");

}



void right_hand_side()
{
        void rlft3(float ***data, float **spec, unsigned long nn1, unsigned long nn2
, unsigned long nn3, int isign);
        int j, k, l, m, n, kk, ll;


        if ((dc==1) || (time<=width)) {

                H_X=Hx;
                H_Y=Hy;
                H_Z=Hz;
        }
        else{
                H_X=0;
                H_Y=0;
                H_Z=0;
        }

        W_ext_at_time_t=0.0;
        W_ani_at_time_t=0.0;
        W_xch_at_time_t=0.0;


        for (kk=1;kk<=MAXROWS;kk++){
                for (ll=1;ll<=MAXCOLS;ll++){
                        if ( (kk>=myrows+1) && (ll>=mycolumns+1) ){
                                M_X_K[1][kk][ll] = M_X[kk-myrows][ll-mycolumns];
                                M_Y_K[1][kk][ll] = M_Y[kk-myrows][ll-mycolumns];
                                M_Z_K[1][kk][ll] = M_Z[kk-myrows][ll-mycolumns];
                        }
                        else{
                                M_X_K[1][kk][ll]=0.0;
                                M_Y_K[1][kk][ll]=0.0;
                                M_Z_K[1][kk][ll]=0.0;
                        }
                }
        }


        for (k=1;k<=myrows;k++){
                for (l=1;l<=mycolumns;l++){
                /*H_ani_X=0;
                H_ani_Y=0;
                H_ani_Z=0;

                  H_xch_X=0;
                  H_xch_Y=0;
                        H_xch_Z=0;*/



                        H_xch_X_total=0.0;
                        H_xch_Y_total=0.0;
                        H_xch_Z_total=0.0;



                        //external magnetic field is already known: H_X, H_Y, and H_
Z


                        //Effective anisotropy field:
                        H_ani_X = 2*(K_u/M)*(M_X[k][l]);
                        H_ani_Y = 0.0;//2*(K_u/M)*M_Y[k][l];
                        H_ani_Z = 0.0;//2*(K_u/M)*M_Z[k][l];
```

```
                        // Effective exchange energy field:

                        for (m=k-1;m<=k+1;m++){
                                for (n=l-1;n<=l+1;n++){

                                        H_xch_X = 0;
                                        H_xch_Y = 0;
                                        H_xch_Z = 0;

                                        if ((m>0)&&(m<=rows) &&(n>0) && (n<=columns) &&
(!((m==k) && (n==l)))){
                                                if ((m+n-k-l==-2) || (m+n-k-l==0) || (m+
n-k-l==2)){

                                                        H_xch_X = 2*A_x*M_X[m][n]/(2*dsq
uare*M);
                                                        H_xch_Y = 2*A_x*M_Y[m][n]/(2*dsq
uare*M);
                                                        H_xch_Z = 2*A_x*M_Z[m][n]/(2*dsq
uare*M);

                                                        // exchange energy contribution:
                                                        if (!( ((M_X[k][l]==0.0) && (M_Y
[k][l]==0.0) && (M_Z[k][l]==0.0)) || ((M_X[m][n]==0.0) && (M_Y[m][n]==0.0) && (M_Z[m][n]
==0.0) )  )){
                                                                W_xch_at_time_t=W_xch_at
_time_t + ( (2*A_x/(2*dsquare))*(1-(M_X[k][l]*M_X[m][n] + M_Y[k][l]*M_Y[m][n] + M_Z[k][l
]*M_Z[m][n])) );}
                                                }
                                                else{

                                                        H_xch_X = 2*A_x*M_X[m][n]/(dsqua
re*M);
                                                        H_xch_Y = 2*A_x*M_Y[m][n]/(dsqua
re*M);
                                                        H_xch_Z = 2*A_x*M_Z[m][n]/(dsqua
re*M);

                                                        // exchange energy contribution:
                                                        if (!( ((M_X[k][l]==0.0) && (M_Y
[k][l]==0.0) && (M_Z[k][l]==0.0)) || ((M_X[m][n]==0.0) && (M_Y[m][n]==0.0) && (M_Z[m][n]
==0.0) )  )){
                                                                W_xch_at_time_t=W_xch_at
_time_t + ( (2*A_x/(dsquare))*(1-(M_X[k][l]*M_X[m][n] + M_Y[k][l]*M_Y[m][n] + M_Z[k][l]*
M_Z[m][n])) );}

                                                }
                                        }

                                        H_xch_X_total=H_xch_X_total+H_xch_X;
                                        H_xch_Y_total=H_xch_Y_total+H_xch_Y;
                                        H_xch_Z_total=H_xch_Z_total+H_xch_Z;
                                }
                        }



                        //the exchange energy sum is over nearest neighbours. The demagn
itizing field sum
                        // is over significant neighbours (i.e. as many as computational
ly practical)

                        H_total_X[k][l] = H_X + H_ani_X + H_xch_X_total;
                        H_total_Y[k][l] = H_Y + H_ani_Y + H_xch_Y_total;
                        H_total_Z[k][l] = H_Z + H_ani_Z + H_xch_Z_total;
                        //
                        // iteratively approximating the LLG equation:
                        //

                        // external field energy contribution:
                        W_ext_at_time_t=W_ext_at_time_t - M*(H_X*M_X[k][l] + H_Y*M_Y
[k][l] + H_Z*M_Z[k][l]);

                        // anisotropy energy contribution:
                        if (!( (M_X[k][l]==0.0) && (M_Y[k][l]==0.0) && (M_Z[k][l]==0
.0) )){
                                W_ani_at_time_t=W_ani_at_time_t + K_u*(1-(M_X[k][l]*
M_X[k][l]));}

                        // exchange energy contribution must be divided by two, to e
liminate double counting:
                        temp_xch=W_xch_at_time_t/2;
                        W_xch_at_time_t=temp_xch;
                }
        }

        rlft3(M_X_K,spec_X,1,MAXROWS,MAXCOLS,1);
        rlft3(M_Y_K,spec_Y,1,MAXROWS,MAXCOLS,1);
        rlft3(M_Z_K,spec_Z,1,MAXROWS,MAXCOLS,1);

        sp1_x = &M_X_K[1][1][1];
        sp2_x = &G_00[1][1][1];
        sp1_y = &M_Y_K[1][1][1];
        sp2_y = &G_01[1][1][1];
        sp1_z = &M_Z_K[1][1][1];
        sp2_z = &G_02[1][1][1];
        sp_Hx = &H_X_K[1][1][1];

        for (j=1; j<=MAXROWS*MAXCOLS/2;j++){
                r_x = sp1_x[0]*sp2_x[0] - sp1_x[1]*sp2_x[1];
                i_x = sp1_x[0]*sp2_x[1] + sp1_x[1]*sp2_x[0];
                r_y = sp1_y[0]*sp2_y[0] - sp1_y[1]*sp2_y[1];
                i_y = sp1_y[0]*sp2_y[1] + sp1_y[1]*sp2_y[0];
                r_z = sp1_z[0]*sp2_z[0] - sp1_z[1]*sp2_z[1];
                i_z = sp1_z[0]*sp2_z[1] + sp1_z[1]*sp2_z[0];

                sp_Hx[0] = fac*(r_x+r_y+r_z);
                sp_Hx[1] = fac*(i_x+i_y+i_z);

                sp1_x +=2;
                sp2_x +=2;
                sp1_y +=2;
                sp2_y +=2;
                sp1_z +=2;
                sp2_z +=2;
                sp_Hx +=2;

        }

        sp1_x = &spec_X[1][1];
        sp2_x = &spec_G_00[1][1];
        sp1_y = &spec_Y[1][1];
        sp2_y = &spec_G_01[1][1];
        sp1_z = &spec_Z[1][1];
        sp2_z = &spec_G_02[1][1];

        sp_Hx = &spec_H_X_K[1][1];
```

```
for (j=1;j<=1*MAXROWS; j++){
        r_x = sp1_x[0]*sp2_x[0] - sp1_x[1]*sp2_x[1];
        i_x = sp1_x[0]*sp2_x[1] + sp1_x[1]*sp2_x[0];
        r_y = sp1_y[0]*sp2_y[0] - sp1_y[1]*sp2_y[1];
        i_y = sp1_y[0]*sp2_y[1] + sp1_y[1]*sp2_y[0];
        r_z = sp1_z[0]*sp2_z[0] - sp1_z[1]*sp2_z[1];
        i_z = sp1_z[0]*sp2_z[1] + sp1_z[1]*sp2_z[0];

        sp_Hx[0] = fac*(r_x+r_y+r_z);
        sp_Hx[1] = fac*(i_x+i_y+i_z);

        sp1_x +=2;
        sp2_x +=2;
        sp1_y +=2;
        sp2_y +=2;
        sp1_z +=2;
        sp2_z +=2;
        sp_Hx +=2;
}



sp1_x = &M_X_K[1][1][1];
sp2_x = &G_10[1][1][1];
sp1_y = &M_Y_K[1][1][1];
sp2_y = &G_11[1][1][1];
sp1_z = &M_Z_K[1][1][1];
sp2_z = &G_12[1][1][1];

sp_Hy = &H_Y_K[1][1][1];

for (j=1; j<=MAXROWS*MAXCOLS/2;j++){
        r_x = sp1_x[0]*sp2_x[0] - sp1_x[1]*sp2_x[1];
        i_x = sp1_x[0]*sp2_x[1] + sp1_x[1]*sp2_x[0];
        r_y = sp1_y[0]*sp2_y[0] - sp1_y[1]*sp2_y[1];
        i_y = sp1_y[0]*sp2_y[1] + sp1_y[1]*sp2_y[0];
        r_z = sp1_z[0]*sp2_z[0] - sp1_z[1]*sp2_z[1];
        i_z = sp1_z[0]*sp2_z[1] + sp1_z[1]*sp2_z[0];

        sp_Hy[0] = fac*(r_x+r_y+r_z);
        sp_Hy[1] = fac*(i_x+i_y+i_z);

        sp1_x +=2;
        sp2_x +=2;
        sp1_y +=2;
        sp2_y +=2;
        sp1_z +=2;
        sp2_z +=2;
        sp_Hy +=2;

}

sp1_x = &spec_X[1][1];
sp2_x = &spec_G_10[1][1];
sp1_y = &spec_Y[1][1];
sp2_y = &spec_G_11[1][1];
sp1_z = &spec_Z[1][1];
sp2_z = &spec_G_12[1][1];

sp_Hy = &spec_H_Y_K[1][1];

for (j=1;j<=1*MAXROWS; j++){
        r_x = sp1_x[0]*sp2_x[0] - sp1_x[1]*sp2_x[1];
        i_x = sp1_x[0]*sp2_x[1] + sp1_x[1]*sp2_x[0];
        r_y = sp1_y[0]*sp2_y[0] - sp1_y[1]*sp2_y[1];
        i_y = sp1_y[0]*sp2_y[1] + sp1_y[1]*sp2_y[0];
        r_z = sp1_z[0]*sp2_z[0] - sp1_z[1]*sp2_z[1];
        i_z = sp1_z[0]*sp2_z[1] + sp1_z[1]*sp2_z[0];

        sp_Hy[0] = fac*(r_x+r_y+r_z);
        sp_Hy[1] = fac*(i_x+i_y+i_z);

        sp1_x +=2;
        sp2_x +=2;
        sp1_y +=2;
        sp2_y +=2;
        sp1_z +=2;
        sp2_z +=2;
        sp_Hy +=2;
}


sp1_x = &M_X_K[1][1][1];
sp2_x = &G_20[1][1][1];
sp1_y = &M_Y_K[1][1][1];
sp2_y = &G_21[1][1][1];
sp1_z = &M_Z_K[1][1][1];
sp2_z = &G_22[1][1][1];

sp_Hz = &H_Z_K[1][1][1];

for (j=1; j<=MAXROWS*MAXCOLS/2;j++){
        r_x = sp1_x[0]*sp2_x[0] - sp1_x[1]*sp2_x[1];
        i_x = sp1_x[0]*sp2_x[1] + sp1_x[1]*sp2_x[0];
        r_y = sp1_y[0]*sp2_y[0] - sp1_y[1]*sp2_y[1];
        i_y = sp1_y[0]*sp2_y[1] + sp1_y[1]*sp2_y[0];
        r_z = sp1_z[0]*sp2_z[0] - sp1_z[1]*sp2_z[1];
        i_z = sp1_z[0]*sp2_z[1] + sp1_z[1]*sp2_z[0];

        sp_Hz[0] = fac*(r_x+r_y+r_z);
        sp_Hz[1] = fac*(i_x+i_y+i_z);

        sp1_x +=2;
        sp2_x +=2;
        sp1_y +=2;
        sp2_y +=2;
        sp1_z +=2;
        sp2_z +=2;
        sp_Hz +=2;

}

sp1_x = &spec_X[1][1];
sp2_x = &spec_G_20[1][1];
sp1_y = &spec_Y[1][1];
sp2_y = &spec_G_21[1][1];
sp1_z = &spec_Z[1][1];
sp2_z = &spec_G_22[1][1];

sp_Hz = &spec_H_Z_K[1][1];

for (j=1;j<=1*MAXROWS; j++){
        r_x = sp1_x[0]*sp2_x[0] - sp1_x[1]*sp2_x[1];
        i_x = sp1_x[0]*sp2_x[1] + sp1_x[1]*sp2_x[0];
        r_y = sp1_y[0]*sp2_y[0] - sp1_y[1]*sp2_y[1];
        i_y = sp1_y[0]*sp2_y[1] + sp1_y[1]*sp2_y[0];
        r_z = sp1_z[0]*sp2_z[0] - sp1_z[1]*sp2_z[1];
        i_z = sp1_z[0]*sp2_z[1] + sp1_z[1]*sp2_z[0];

        sp_Hz[0] = fac*(r_x+r_y+r_z);
```

```
                sp_Hz[1] = fac*(i_x+i_y+i_z);

                sp1_x +=2;
                sp2_x +=2;
                sp1_y +=2;
                sp2_y +=2;
                sp1_z +=2;
                sp2_z +=2;
                sp_Hz +=2;

        }

        rlft3(H_X_K,spec_H_X_K,1,MAXROWS,MAXCOLS,-1);
        rlft3(H_Y_K,spec_H_Y_K,1,MAXROWS,MAXCOLS,-1);
        rlft3(H_Z_K,spec_H_Z_K,1,MAXROWS,MAXCOLS,-1);




}



// the gateway function

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{


        //llginc(alpha, gamma, d, M, K_u, A_x, rows, columns, time_step, time_limit, Hx,
 Hy, Hz, dc, width, finish)


        double *modifiedtime, *W_ext_pt, *W_ani_pt, *W_xch_pt, *points_X_pt, *points_Y_p
t, *points_Z_pt, *X_pt, *Y_pt, *Z_pt, *points_aveX_pt, *points_aveY_pt, *points_aveZ_pt;



        alpha=mxGetScalar(prhs[0]);
        gamma=mxGetScalar(prhs[1]);
        d=mxGetScalar(prhs[2]);
        M=mxGetScalar(prhs[3]);
        K_u=mxGetScalar(prhs[4]);
        A_x=mxGetScalar(prhs[5]);
        rows=mxGetScalar(prhs[6]);
        columns=mxGetScalar(prhs[7]);
        time_step=mxGetScalar(prhs[8]);
        time_limit=mxGetScalar(prhs[9]);
        Hx=mxGetScalar(prhs[10]);
        Hy=mxGetScalar(prhs[11]);
        Hz=mxGetScalar(prhs[12]);
        dc=mxGetScalar(prhs[13]);
        width=mxGetScalar(prhs[14]);
        finish=mxGetScalar(prhs[15]);
        displayfreq=mxGetScalar(prhs[16]);
        buflen = (mxGetM(prhs[17]) * mxGetN(prhs[17])) + 1;
        foome_x = mxGetPr(prhs[18]);
        foome_y = mxGetPr(prhs[19]);
        foome_z = mxGetPr(prhs[20]);
        toleranceX=mxGetScalar(prhs[21]);
        replenishtime_limit=mxGetScalar(prhs[22]);
        h=mxGetScalar(prhs[23]);

        displayfrequency=(int)displayfreq;
        buf = mxCalloc(buflen, sizeof(char));
        mxGetString(prhs[17], buf, buflen);

        plhs[0]=mxCreateDoubleMatrix((int)rows, (int)columns, mxREAL);
        plhs[1]=mxCreateDoubleMatrix((int)rows, (int)columns, mxREAL);
        plhs[2]=mxCreateDoubleMatrix((int)rows, (int)columns, mxREAL);

        plhs[3]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);
        plhs[4]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);
        plhs[5]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);
        plhs[6]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);
        plhs[7]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);
        plhs[8]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);
        plhs[9]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);
        plhs[10]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);
        plhs[11]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);
        plhs[12]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);

        plhs[13]=mxCreateDoubleMatrix(1,1,mxREAL);
        plhs[14]=mxCreateDoubleMatrix(1,1,mxREAL);
        plhs[15]=mxCreateDoubleMatrix(1,1,mxREAL);

        plhs[16]=mxCreateDoubleMatrix(1, (int)(time_limit/time_step)+600, mxREAL);



        X_pt=mxGetPr(plhs[0]);
        Y_pt=mxGetPr(plhs[1]);
        Z_pt=mxGetPr(plhs[2]);
        points_X_pt=mxGetPr(plhs[3]);
        points_Y_pt=mxGetPr(plhs[4]);
        points_Z_pt=mxGetPr(plhs[5]);
        W_ext_pt=mxGetPr(plhs[6]);
        W_ani_pt=mxGetPr(plhs[7]);
        W_xch_pt=mxGetPr(plhs[8]);
        modifiedtime=mxGetPr(plhs[9]);
        points_aveX_pt=mxGetPr(plhs[10]);
        points_aveY_pt=mxGetPr(plhs[11]);
        points_aveZ_pt=mxGetPr(plhs[12]);

        meanmeanMX=mxGetPr(plhs[13]);
        meanmeanMY=mxGetPr(plhs[14]);
        meanmeanMZ=mxGetPr(plhs[15]);

        totalenergy_pt=mxGetPr(plhs[16]);


        // call the computational routine
        myllg(X_pt, Y_pt, Z_pt, points_X_pt, points_Y_pt, points_Z_pt, W_ext_pt, W_a
ni_pt, W_xch_pt, modifiedtime, points_aveX_pt, points_aveY_pt, points_aveZ_pt);



}
```

```
void rlft3(float ***data, float **speq, unsigned long nn1, unsigned long nn2,
                  unsigned long nn3, int isign)
{
        void fourn(float data[], unsigned long nn[], int ndim, int isign);
        void nrerror(char error_text[]);
        unsigned long i1,i2,i3,j1,j2,j3,nn[4],ii3;
        double theta,wi,wpi,wpr,wr,wtemp;
        float c1,c2,h1r,h1i,h2r,h2i;

        if (1+&data[nn1][nn2][nn3]-&data[1][1][1] != nn1*nn2*nn3)
                nrerror("rlft3: problem with dimensions or contiguity of data array\n");
        c1=0.5;
        c2 = -0.5*isign;
        theta=isign*(6.28318530717959/nn3);
        wtemp=sin(0.5*theta);
        wpr = -2.0*wtemp*wtemp;
        wpi=sin(theta);
        nn[1]=nn1;
        nn[2]=nn2;
        nn[3]=nn3 >> 1;
        if (isign == 1) {
                fourn(&data[1][1][1]-1,nn,3,isign);
                for (i1=1;i1<=nn1;i1++)
                        for (i2=1,j2=0;i2<=nn2;i2++) {
                                speq[i1][++j2]=data[i1][i2][1];
                                speq[i1][++j2]=data[i1][i2][2];
                        }
        }
        for (i1=1;i1<=nn1;i1++) {
                j1=(i1 != 1 ? nn1-i1+2 : 1);
                wr=1.0;
                wi=0.0;
                for (ii3=1,i3=1;i3<=(nn3>>2)+1;i3++,ii3+=2) {
                        for (i2=1;i2<=nn2;i2++) {
                                if (i3 == 1) {
                                        j2=(i2 != 1 ? ((nn2-i2)<<1)+3 : 1);
                                        h1r=c1*(data[i1][i2][1]+speq[j1][j2]);
                                        h1i=c1*(data[i1][i2][2]-speq[j1][j2+1]);
                                        h2i=c2*(data[i1][i2][1]-speq[j1][j2]);
                                        h2r= -c2*(data[i1][i2][2]+speq[j1][j2+1]);
                                        data[i1][i2][1]=h1r+h2r;
                                        data[i1][i2][2]=h1i+h2i;
                                        speq[j1][j2]=h1r-h2r;
                                        speq[j1][j2+1]=h2i-h1i;
                                } else {
                                        j2=(i2 != 1 ? nn2-i2+2 : 1);
                                        j3=nn3+3-(i3<<1);
                                        h1r=c1*(data[i1][i2][ii3]+data[j1][j2][j3]);
                                        h1i=c1*(data[i1][i2][ii3+1]-data[j1][j2][j3+1]);
                                        h2i=c2*(data[i1][i2][ii3]-data[j1][j2][j3]);
                                        h2r= -c2*(data[i1][i2][ii3+1]+data[j1][j2][j3+1]
);
                                        data[i1][i2][ii3]=h1r+wr*h2r-wi*h2i;
                                        data[i1][i2][ii3+1]=h1i+wr*h2i+wi*h2r;
                                        data[j1][j2][j3]=h1r-wr*h2r+wi*h2i;
                                        data[j1][j2][j3+1]= -h1i+wr*h2i+wi*h2r;
                                }
                        }
                        wr=(wtemp=wr)*wpr-wi*wpi+wr;
                        wi=wi*wpr+wtemp*wpi+wi;
                }
        }
        if (isign == -1)
                fourn(&data[1][1][1]-1,nn,3,isign);
}
/* (C) Copr. 1986-92 Numerical Recipes Software *1.@(1.35.. */



void fourn(float data[], unsigned long nn[], int ndim, int isign)
{
        int idim;
        unsigned long i1,i2,i3,i2rev,i3rev,ip1,ip2,ip3,ifp1,ifp2;
        unsigned long ibit,k1,k2,n,nprev,nrem,ntot;
        float tempi,tempr;
        double theta,wi,wpi,wpr,wr,wtemp;

        for (ntot=1,idim=1;idim<=ndim;idim++)
                ntot *= nn[idim];
        nprev=1;
        for (idim=ndim;idim>=1;idim--) {
                n=nn[idim];
                nrem=ntot/(n*nprev);
                ip1=nprev << 1;
                ip2=ip1*n;
                ip3=ip2*nrem;
                i2rev=1;
                for (i2=1;i2<=ip2;i2+=ip1) {
                        if (i2 < i2rev) {
                                for (i1=i2;i1<=i2+ip1-2;i1+=2) {
                                        for (i3=i1;i3<=ip3;i3+=ip2) {
                                                i3rev=i2rev+i3-i2;
                                                SWAP(data[i3],data[i3rev]);
                                                SWAP(data[i3+1],data[i3rev+1]);
                                        }
                                }
                        }
                        ibit=ip2 >> 1;
                        while (ibit >= ip1 && i2rev > ibit) {
                                i2rev -= ibit;
                                ibit >>= 1;
                        }
                        i2rev += ibit;
                }
                ifp1=ip1;
                while (ifp1 < ip2) {
                        ifp2=ifp1 << 1;
                        theta=isign*6.28318530717959/(ifp2/ip1);
                        wtemp=sin(0.5*theta);
                        wpr = -2.0*wtemp*wtemp;
                        wpi=sin(theta);
                        wr=1.0;
                        wi=0.0;
                        for (i3=1;i3<=ifp1;i3+=ip1) {
                                for (i1=i3;i1<=i3+ip1-2;i1+=2) {
                                        for (i2=i1;i2<=ip3;i2+=ifp2) {
                                                k1=i2;
                                                k2=k1+ifp1;
                                                tempr=(float)wr*data[k2]-(float)wi*d
ata[k2+1];
                                                tempi=(float)wr*data[k2+1]+(float)wi
*data[k2];
                                                data[k2]=data[k1]-tempr;
                                                data[k2+1]=data[k1+1]-tempi;
                                                data[k1] += tempr;
```

```
                                        data[k1+1] += tempi;
                                    }
                                }
                                wr=(wtemp=wr)*wpr-wi*wpi+wr;
                                wi=wi*wpr+wtemp*wpi+wi;
                            }
                            ifp1=ifp2;
                        }
                        nprev *= n;
                    }
}
/* (C) Copr. 1986-92 Numerical Recipes Software *1.@(1.35.. */
```

```
function fig = createmovie()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.

load createmovie

h0 = figure('Color',[0.8 0.8 0.8], ...
        'Colormap',mat0, ...
        'PointerShapeCData',mat1, ...
        'Position',[215 152 771 561], ...
        'Tag','Fig1');
h1 = axes('Parent',h0, ...
        'CameraUpVector',[0 1 0], ...
        'CameraUpVectorMode','manual', ...
        'Color',[1 1 1], ...
        'ColorOrder',mat2, ...
        'Position',[0.14786 0.253119 0.688716 0.643494], ...
        'Tag','Axes1', ...
        'XColor',[0 0 0], ...
        'XLim',[1 32], ...
        'XLimMode','manual', ...
        'YColor',[0 0 0], ...
        'YLim',[-3.2 32], ...
        'YLimMode','manual', ...
        'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[16.4415 -5.54667 0], ...
        'Tag','Axes1Text4', ...
        'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[-0.462264 14.3022 0], ...
        'Rotation',90, ...
        'Tag','Axes1Text3', ...
        'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Position',[-5.72642 37.5733 0], ...
        'Tag','Axes1Text2', ...
        'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[16.4415 32.6844 0], ...
        'Tag','Axes1Text1', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat3, ...
        'ListboxTop',0, ...
        'Position',[15.75 18.75 57 15], ...
        'String','Create Movie', ...
        'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback','movie(MM,3);', ...
        'ListboxTop',0, ...
        'Position',[15 396.75 84.75 15], ...
        'String','Play MATLAB Movie', ...
        'Tag','Pushbutton2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat4, ...
        'ListboxTop',0, ...
        'Position',[17.25 48.75 73.5 19.5], ...
        'String','Save to MPEG File', ...
        'Tag','Pushbutton3');
h1 = axes('Parent',h0, ...
        'Box','on', ...
        'CameraUpVector',[0 1 0], ...
        'CameraUpVectorMode','manual', ...
        'CLim',[0 6.28319], ...
        'CLimMode','manual', ...
        'Color',[1 1 1], ...
        'ColorOrder',mat5, ...
        'Layer','top', ...
        'Position',[0.8884570000000001 0.768271 0.0843061 0.124777], ...
        'Tag','Axes2', ...
        'Visible','off', ...
        'XColor',[0 0 0], ...
        'XLim',[0.5 201.5], ...
        'XLimMode','manual', ...
        'YColor',[0 0 0], ...
        'YDir','reverse', ...
        'YLim',[0.5 201.5], ...
        'YLimMode','manual', ...
        'ZColor',[0 0 0]);
h2 = image('Parent',h1, ...
        'CData',mat6, ...
        'CDataMapping','scaled', ...
        'Tag','Image1', ...
        'XData',[1 201], ...
        'YData',[1 201]);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[97.85939999999999 272.441 0], ...
        'Tag','Text4', ...
        'VerticalAlignment','cap', ...
        'Visible','off');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[-100 103.956 0], ...
        'Rotation',90, ...
        'Tag','Text3', ...
        'VerticalAlignment','baseline', ...
        'Visible','off');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
```

```
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Position',[-2153.97 -173.897 0], ...
        'Tag','Text2', ...
        'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[97.85939999999999 -20.1912 0], ...
        'Tag','Text1', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[289.5 9 45 15], ...
        'String','10', ...
        'Style','edit', ...
        'Tag','EditText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[290.25 21 45 15], ...
        'String','# of frames', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[100.5 10.5 179.25 15], ...
        'String','D:\HUR_LLG_Simulator\Movies\eraseme.txt', ...
        'Style','edit', ...
        'Tag','EditText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[117 24.75 87 15], ...
        'String','Read Movie File From', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[8.0625 243 33 15], ...
        'String','fill in', ...
        'Style','edit', ...
        'Tag','EditText3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[8.4375 201 32.25 15], ...
        'String','fill in', ...
        'Style','edit', ...
        'Tag','EditText4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[8.4375 258 32.25 12], ...
        'String','Rows', ...
        'Style','text', ...
        'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[3 216 45 12], ...
        'String','Columns', ...
        'Style','text', ...
        'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[99 45.75 179.25 15], ...
        'String','D:\HUR_LLG_Simulator\Movies\eraseme.mpg', ...
        'Style','edit', ...
        'Tag','EditText5');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[119.25 60.75 86.25 15], ...
        'String','MPEG File to Write', ...
        'Style','text', ...
        'Tag','StaticText5');
if nargout > 0, fig = h0; end
```

```
function fig = loop()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.

load loop

h0 = figure('Color',[0.8 0.8 0.8], ...
        'Colormap',mat0, ...
        'CreateFcn','set(gcf,''name'',''Loop Control Window'');', ...
        'Name','Loop Control Window', ...
        'PointerShapeCData',mat1, ...
        'Position',[739 367 222 321], ...
        'Tag','Fig2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[13.5 198 54.75 15], ...
        'String','-300', ...
        'Style','edit', ...
        'Tag','EditText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[13.5 156 54 14.25], ...
        'String','-1000', ...
        'Style','edit', ...
        'Tag','EditText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[13.5 120 54 15], ...
        'String','25', ...
        'Style','edit', ...
        'Tag','EditText3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[94.5 198 55.5 15], ...
        'String','-30', ...
        'Style','edit', ...
        'Tag','EditText4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[9 213 63 23.25], ...
        'String','Starting X Field in Loop (Oe)', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[7.5 171.75 66.75 23.25], ...
        'String','Ending X Field in Loop (Oe)', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[14.25 132 52.5 22.5], ...
        'String','X Field Increment (Oe)', ...
        'Style','text', ...
        'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[96 155.25 54 15], ...
        'String','-230', ...
        'Style','edit', ...
        'Tag','EditText5');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[95.25 120 54.75 15], ...
        'String','-20', ...
        'Style','edit', ...
        'Tag','EditText6');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[92.25 213 58.5 23.25], ...
        'String','Starting Y-Field in Loop (Oe)', ...
        'Style','text', ...
        'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[94.5 169.5 57 25.5], ...
        'String','Ending Y-Field in Loop (Oe)', ...
        'Style','text', ...
        'Tag','StaticText5');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[95.25 132.75 55.5 21], ...
        'String','Y-Field Increment (Oe)', ...
        'Style','text', ...
        'Tag','StaticText6');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat2, ...
        'ListboxTop',0, ...
        'Position',[16.5 47.25 55.5 15], ...
        'String','START LOOP', ...
        'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Callback','set(findobj(gcf,''Tag'',''Checkbox2''),''Value'',0) ;', ...
        'ListboxTop',0, ...
        'Position',[15.75 99 128.25 15], ...
        'String','Reset Back to Starting X-Field?', ...
        'Style','checkbox', ...
        'Tag','Checkbox1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
```

```
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[116.25 66.75 45 15], ...
        'String','-250', ...
        'Style','edit', ...
        'Tag','EditText7');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[117 45 45 15], ...
        'String','0', ...
        'Style','edit', ...
        'Tag','EditText8');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[80.25 66 36.75 13.5], ...
        'String','X-Reset', ...
        'Style','text', ...
        'Tag','StaticText7');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[79.5 45 37.5 14.25], ...
        'String','Y-Reset', ...
        'Style','text', ...
        'Tag','StaticText8');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Callback','set(findobj(gcf,''Tag'',''Checkbox1''),''Value'',0) ;', ...
        'ListboxTop',0, ...
        'Position',[15.75 82.5 129 18.75], ...
        'String','Use Starting Positiions?', ...
        'Style','checkbox', ...
        'Tag','Checkbox2', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat3, ...
        'ListboxTop',0, ...
        'Position',[16.5 24.75 51.75 15], ...
        'String','ASTEROID', ...
        'Tag','Pushbutton2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[117 12.75 45 15], ...
        'String','8', ...
        'Style','edit', ...
        'Tag','EditText9');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[116.25 27.75 45 15], ...
        'String','# of iter.', ...
        'Style','text', ...
        'Tag','StaticText9');
if nargout > 0, fig = h0; end
```

plotfield4.m


```
angles = atan2(M_Y,M_X);

size_angles = size(angles);
for k = 1:size_angles(1),
   for l = 1:size_angles(2),
      if angles(k,l)<0
         angles(k,l)=angles(k,l)+2*pi;
      end;
   end;
end;
%colormap(gray);
imagesc(angles,[0 2*pi]);
```

llgfigurenew.m


```
function fig = llgfigurenew()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.

load llgfigurenew

h0 = figure('Color',[0.8 0.8 0.8], ...
        'Colormap',mat0, ...
        'CreateFcn','set(gcf,''name'',''LLG Integrator'');IntegratorHandle=gcf;', ...
        'Name','LLG Integrator', ...
        'PointerShapeCData',mat1, ...
        'Position',[137 102 842 633], ...
        'Tag','Fig1', ...
        'WindowButtonDownFcn','rotate3d(''down'')', ...
        'WindowButtonUpFcn','rotate3d(''up'')');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback','LLGfaster;', ...
        'ListboxTop',0, ...
        'Position',[8.25 447.75 45 15], ...
        'String','Start', ...
        'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback','finish=1;', ...
        'ListboxTop',0, ...
        'Position',[7.5 15 45 15], ...
        'String','STOP', ...
        'Tag','Pushbutton2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[8.25 404.25 45 15], ...
        'String','0.01', ...
        'Style','edit', ...
        'Tag','EditText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[8.25 368.25 42.75 15.75], ...
        'String','17.6e6', ...
        'Style','edit', ...
        'Tag','EditText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[8.25 333.75 45 15], ...
        'String','5e-7', ...
        'Style','edit', ...
        'Tag','EditText3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[9.75 273.75 45 15], ...
        'String','800', ...
        'Style','edit', ...
        'Tag','EditText4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[9.75 240.75 45 15], ...
        'String','1000', ...
        'Style','edit', ...
        'Tag','EditText5');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[9.75 206.25 45 15], ...
        'String','1e-6', ...
        'Style','edit', ...
        'Tag','EditText6');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat2, ...
        'ListboxTop',0, ...
        'Position',[75 441 45 15], ...
        'String','128', ...
        'Style','edit', ...
        'Tag','EditText7');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat3, ...
        'ListboxTop',0, ...
        'Position',[74.25 410.25 45 15], ...
        'String','64', ...
        'Style','edit', ...
        'Tag','EditText8');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[75 375.75 45 15], ...
        'String','0.8e-12', ...
        'Style','edit', ...
        'Tag','EditText9');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[75.75 345 45 15], ...
        'String','1000e-12', ...
        'Style','edit', ...
        'Tag','EditText10');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[75.75 267.75 45 15], ...
        'String','50e-12', ...
        'Style','edit', ...
        'Tag','EditText11', ...
        'Visible','off');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[75.75 216 44.25 16.5], ...
```

```
        'String','-200', ...
        'Style','edit', ...
        'Tag','EditText12');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[75.75 188.25 45 15], ...
        'String','-10', ...
        'Style','edit', ...
        'Tag','EditText13');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[75 159.75 45 15], ...
        'String','0', ...
        'Style','edit', ...
        'Tag','EditText14');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Callback','set(findobj(gcf,''Tag'',''Radiobutton2''),''Value'',0);', ...
        'ListboxTop',0, ...
        'Position',[9.75 185.25 56.25 15], ...
        'String','Forw. Euler', ...
        'Style','radiobutton', ...
        'Tag','Radiobutton1', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Callback','set(findobj(gcf,''Tag'',''Radiobutton1''),''Value'',0);', ...
        'ListboxTop',0, ...
        'Position',[9 170.25 59.25 15], ...
        'String','Runga-Kutta', ...
        'Style','radiobutton', ...
        'Tag','Radiobutton2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Callback',mat4, ...
        'ListboxTop',0, ...
        'Position',[75.75 295.5 45 15], ...
        'String','DC Field', ...
        'Style','radiobutton', ...
        'Tag','Radiobutton3', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat5, ...
        'ListboxTop',0, ...
        'Position',[75.75 284.25 45 15], ...
        'String','Pulse', ...
        'Style','radiobutton', ...
        'Tag','Radiobutton4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[8.25 418.5 45 15], ...
        'String','Alpha', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[8.25 384 45 15], ...
        'String','Gamma', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[8.25 348.75 45 15], ...
        'String','d', ...
        'Style','text', ...
        'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[9.75 288.75 45 15], ...
        'String','M', ...
        'Style','text', ...
        'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[9 255 45 15], ...
        'String','K_u', ...
        'Style','text', ...
        'Tag','StaticText5');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[9.75 222 45 15], ...
        'String','A_x', ...
        'Style','text', ...
        'Tag','StaticText6');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[75 455.25 45 15], ...
        'String','# of rows', ...
        'Style','text', ...
        'Tag','StaticText7');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[75 425.25 49.5 11.25], ...
        'String','# of columns', ...
        'Style','text', ...
        'Tag','StaticText8');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[75 390 49.5 15], ...
        'String','Time Step (s)', ...
        'Style','text', ...
        'Tag','StaticText9');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[76.5 359.25 45 15], ...
        'String','End Time (s)', ...
        'Style','text', ...
        'Tag','StaticText10');
h1 = uicontrol('Parent',h0, ...
```

```
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[120 268.5 36 15.75], ...
        'String','Width (s)', ...
        'Style','text', ...
        'Tag','StaticText11', ...
        'Visible','off');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[75.75 232.5 45 15], ...
        'String','Hx (Oe)', ...
        'Style','text', ...
        'Tag','StaticText12');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[75.75 202.5 45 15], ...
        'String','Hy (Oe)', ...
        'Style','text', ...
        'Tag','StaticText13');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'ListboxTop',0, ...
        'Position',[75 173.25 45 15], ...
        'String','Hz (Oe)', ...
        'Style','text', ...
        'Tag','StaticText14');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[9.75 123.75 45 15], ...
        'String','10', ...
        'Style','edit', ...
        'Tag','EditText15');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[10.5 138.75 45 28.5], ...
        'String','Progress Display Frequency', ...
        'Style','text', ...
        'Tag','StaticText15');
h1 = axes('Parent',h0, ...
        'View',[-37.5 30], ...
        'CameraUpVector',[0 0 1], ...
        'CameraUpVectorMode','manual', ...
        'Color',[1 1 1], ...
        'ColorOrder',mat6, ...
        'Position',[0.3 0.55 0.3 0.4], ...
        'Tag','Axes3', ...
        'XColor',[0 0 0], ...
        'XGrid','on', ...
        'XLim',[0 17], ...
        'XLimMode','manual', ...
        'YColor',[0 0 0], ...
        'YGrid','on', ...
        'YLim',[0 17], ...
        'YLimMode','manual', ...
        'ZColor',[0 0 0], ...
        'ZGrid','on', ...
        'ZLim',[-0.3 0.3], ...
        'ZLimMode','manual');
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'Position',[-4.86658 -16.6885 0], ...
        'Tag','Axes3Text4', ...
        'VerticalAlignment','top');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Position',[-13.957 -8.64716 0], ...
        'Tag','Axes3Text3', ...
        'VerticalAlignment','top');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[-2.88437 18.8346 0], ...
        'Rotation',90, ...
        'Tag','Axes3Text2', ...
        'VerticalAlignment','baseline');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[25.5436 30.867 0], ...
        'Tag','Axes3Text1', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = axes('Parent',h0, ...
        'View',[322.5 30], ...
        'CameraUpVector',[0 0 1], ...
        'CameraUpVectorMode','manual', ...
        'Color',[1 1 1], ...
        'ColorOrder',mat7, ...
        'Position',[0.6472684085510689 0.5513428120063191 0.2992874109263658 0.39968
40442338073], ...
        'Tag','Axes2', ...
        'XColor',[0 0 0], ...
        'XGrid','on', ...
        'XLim',[-1 1], ...
        'XLimMode','manual', ...
        'YColor',[0 0 0], ...
        'YGrid','on', ...
        'YLim',[-1 1], ...
        'YLimMode','manual', ...
        'ZColor',[0 0 0], ...
        'ZGrid','on', ...
        'ZLim',[-1 1], ...
        'ZLimMode','manual');
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'Position',mat8, ...
        'Tag','Axes2Text4', ...
        'VerticalAlignment','top');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Position',[-10.30669133072029 -12.00963116303244 7.270807830735933], ...
        'Tag','Axes2Text3', ...
```

```
        'VerticalAlignment','top');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',mat9, ...
        'Rotation',90, ...
        'Tag','Axes2Text2', ...
        'VerticalAlignment','baseline');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',mat10, ...
        'Tag','Axes2Text1', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Callback',mat11, ...
        'ListboxTop',0, ...
        'Min',0.05, ...
        'Position',[135.75 445.5 52.5 9], ...
        'Style','slider', ...
        'Tag','Slider1', ...
        'Value',0.20865);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Callback',mat12, ...
        'ListboxTop',0, ...
        'Min',0.05, ...
        'Position',[124.5 243 54 8.25], ...
        'Style','slider', ...
        'Tag','Slider2', ...
        'Value',0.3825);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[11.25 90 120.75 15], ...
        'Style','edit', ...
        'Tag','EditText16', ...
        'Visible','off');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Callback',mat13, ...
        'ListboxTop',0, ...
        'Position',[15 102.75 108 15], ...
        'String','Want to create movie file?', ...
        'Style','radiobutton', ...
        'Tag','Radiobutton5');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[138.75 454.5 45 15], ...
        'String','X-Zoom', ...
        'Style','text', ...
        'Tag','StaticText17');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[127.5 250.5 45 15], ...
        'String','Y-Zoom', ...
        'Style','text', ...
        'Tag','StaticText18');
h1 = axes('Parent',h0, ...
        'Box','on', ...
        'CameraUpVector',[0 1 0], ...
        'CameraUpVectorMode','manual', ...
        'Color',[1 1 1], ...
        'ColorOrder',mat14, ...
        'Position',[0.276722 0.06793050000000001 0.448931 0.399684], ...
        'Tag','Axes1', ...
        'XColor',[0 0 0], ...
        'YColor',[0 0 0], ...
        'ZColor',[0 0 0]);
h2 = line('Parent',h1, ...
        'Color',[0 0 1], ...
        'Tag','Axes1Line3', ...
        'XData',mat15, ...
        'YData',mat16);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[4.986737400530504e-013 -191235.059760957 17.32050807568877], ...
        'String','Time (s)', ...
        'Tag','Axes1Text4', ...
        'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[-7.692307692307688e-014 988047.8087649397 17.32050807568877], ..
.
        'Rotation',90, ...
        'String','Energy ergs', ...
        'Tag','Axes1Text3', ...
        'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = line('Parent',h1, ...
        'Color',[0 0 0], ...
        'Tag','Axes1Line2', ...
        'XData',mat17, ...
        'YData',mat18);
h2 = line('Parent',h1, ...
        'Color',[1 0 0], ...
        'Tag','Axes1Line1', ...
        'XData',mat19, ...
        'YData',mat20);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Position',[-6.2069e-013 4666670 0], ...
        'Tag','Axes1Text2', ...
        'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
```

```
        'Position',[4.98674e-013 2055560 0], ...
        'Tag','Axes1Text1', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = axes('Parent',h0, ...
        'Box','on', ...
        'ButtonDownFcn','moveaxis', ...
        'CameraUpVector',[0 1 0], ...
        'CameraUpVectorMode','manual', ...
        'CLimMode','manual', ...
        'Color',[1 1 1], ...
        'ColorOrder',mat21, ...
        'DrawMode','fast', ...
        'NextPlot','add', ...
        'Position',[0.761283 0.317536 0.22209 0.101106], ...
        'Tag','legend', ...
        'UserData',mat22, ...
        'XColor',[0 0 0], ...
        'XLimMode','manual', ...
        'XTick',-1, ...
        'XTickLabelMode','manual', ...
        'XTickMode','manual', ...
        'YColor',[0 0 0], ...
        'YLimMode','manual', ...
        'YTick',-1, ...
        'YTickLabelMode','manual', ...
        'YTickMode','manual', ...
        'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
        'FontUnits','normalized', ...
        'ButtonDownFcn','moveaxis', ...
        'Color',[0 0 0], ...
        'FontSize',0.233918, ...
        'Position',[0.474785 0.75 0], ...
        'String','Ext. H Energy ', ...
        'Tag','legendText7');
h2 = line('Parent',h1, ...
        'ButtonDownFcn','moveaxis', ...
        'Color',[0 0 1], ...
        'Tag','legendLine3', ...
        'XData',[0.06330470000000001 0.316524], ...
        'YData',[0.75 0.75]);
h2 = text('Parent',h1, ...
        'FontUnits','normalized', ...
        'ButtonDownFcn','moveaxis', ...
        'Color',[0 0 0], ...
        'FontSize',0.233918, ...
        'Position',[0.474785 0.5 0], ...
        'String','Anisotropy En.', ...
        'Tag','legendText6');
h2 = line('Parent',h1, ...
        'ButtonDownFcn','moveaxis', ...
        'Color',[0 0 0], ...
        'Tag','legendLine2', ...
        'XData',[0.06330470000000001 0.316524], ...
        'YData',[0.5 0.5]);
h2 = text('Parent',h1, ...
        'FontUnits','normalized', ...
        'ButtonDownFcn','moveaxis', ...
        'Color',[0 0 0], ...
        'FontSize',0.233918, ...
        'Position',[0.474785 0.25 0], ...
        'String','Exchange En.  ', ...
        'Tag','legendText5');
h2 = line('Parent',h1, ...
        'ButtonDownFcn','moveaxis', ...
        'Color',[1 0 0], ...
        'Tag','legendLine1', ...
        'XData',[0.06330470000000001 0.316524], ...
        'YData',[0.25 0.25]);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[0.494624 -0.126984 0], ...
        'Tag','legendText4', ...
        'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[-0.0376344 0.47619 0], ...
        'Rotation',90, ...
        'Tag','legendText3', ...
        'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Position',[-3.45161 6.8254 0], ...
        'Tag','legendText2', ...
        'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[0.494624 1.11111 0], ...
        'Tag','legendText1', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = uicontrol('Parent',h0, ...
        'HandleVisibility','off', ...
        'ListboxTop',0, ...
        'Position',[2 2 130 20], ...
        'Style','text', ...
        'Tag','StaticText16', ...
        'Visible','off');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat23, ...
        'ListboxTop',0, ...
        'Position',[71.25 15 53.25 15], ...
        'String','New Shape', ...
        'Tag','Pushbutton3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'Callback',mat24, ...
        'CreateFcn',mat25, ...
        'ListboxTop',0, ...
        'Position',[474 105 130.5 15], ...
        'String','Nothing Selected', ...
        'Style','popupmenu', ...
        'Tag','PopupMenu1', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
```

```
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[498.75 120.75 81 21], ...
        'String','Choose a predefined shape', ...
        'Style','text', ...
        'Tag','StaticText19');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[490.125 72 98.25 15], ...
        'Style','edit', ...
        'Tag','EditText17');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat26, ...
        'ListboxTop',0, ...
        'Position',[485.25 56.25 48 15], ...
        'String','Save Shape', ...
        'Tag','Pushbutton4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[491.25 87.75 96 13.5], ...
        'String','Save this new shape as:', ...
        'Style','text', ...
        'Tag','StaticText20');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat27, ...
        'ListboxTop',0, ...
        'Position',[541.5 56.25 51 15], ...
        'String','Save System', ...
        'Tag','Pushbutton5');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[73.5 119.25 45 15], ...
        'String','0.0001', ...
        'Style','edit', ...
        'Tag','EditText18');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontSize',4, ...
        'ListboxTop',0, ...
        'Position',[69 133.5 60 24], ...
        'String','Integration Tolerance', ...
        'Style','text', ...
        'Tag','StaticText21');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback','loop;', ...
        'ListboxTop',0, ...
        'Position',[72 36.75 54 15], ...
        'String','LOOP', ...
        'Tag','Pushbutton6');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[497.25 8.25 51 15], ...
        'String','0.0', ...
        'Style','edit', ...
        'Tag','EditText19');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[501 22.5 45 15], ...
        'String','Time', ...
        'Style','text', ...
        'Tag','StaticText22');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[552.75 8.25 60 15], ...
        'String','0.0', ...
        'Style','edit', ...
        'Tag','EditText20');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[559.5 22.5 45 15], ...
        'String','Average M', ...
        'Style','text', ...
        'Tag','StaticText23');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback','createmovie;', ...
        'ListboxTop',0, ...
        'Position',[9 36.75 53.25 15], ...
        'String','Create Movie', ...
        'Tag','Pushbutton7');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[75.75 309.75 45 15.75], ...
        'String','300e-12', ...
        'Style','edit', ...
        'Tag','EditText21');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'ListboxTop',0, ...
        'Position',[69 324.75 57.75 15], ...
        'String','Replenish Time', ...
        'Style','text', ...
        'Tag','StaticText24');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[8.25 303 42 15], ...
        'String','5e-007', ...
        'Style','edit', ...
        'Tag','EditText22');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], .
..
        'ListboxTop',0, ...
        'Position',[8.25 319.5 45 15], ...
        'String','h', ...
        'Style','text', ...
```

```
        'Tag','StaticText25');
if nargout > 0, fig = h0; end
```

```
function fig = initialize()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.

load initialize

h0 = figure('Color',[0.8 0.8 0.8], ...
        'Colormap',mat0, ...
        'PointerShapeCData',mat1, ...
        'Position',[211 88 787 621], ...
        'Renderer','zbuffer', ...
        'RendererMode','manual', ...
        'Tag','Fig1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat2, ...
        'ListboxTop',0, ...
        'Position',[11.25 438.75 54.75 15], ...
        'String','Define Shape', ...
        'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat3, ...
        'ListboxTop',0, ...
        'Position',[11.25 57 58.5 15], ...
        'String','Finish Defining', ...
        'Tag','Pushbutton2');
h1 = axes('Parent',h0, ...
        'Box','on', ...
        'CameraUpVector',[0 1 0], ...
        'CameraUpVectorMode','manual', ...
        'Color',[1 1 1], ...
        'ColorOrder',mat4, ...
        'Position',[0.166455 0.112721 0.7941550000000001 0.816425], ...
        'Tag','Axes1', ...
        'XColor',[0 0 0], ...
        'XLim',[1 17], ...
        'XLimMode','manual', ...
        'YColor',[0 0 0], ...
        'YLim',[1 17], ...
        'YLimMode','manual', ...
        'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[8.974360000000001 0.241107 0], ...
        'Tag','Axes1Text4', ...
        'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[0.358974 8.96838 0], ...
        'Rotation',90, ...
        'Tag','Axes1Text3', ...
        'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Position',[-2.38462 18.3597 0], ...
        'Tag','Axes1Text2', ...
        'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[8.974360000000001 17.2213 0], ...
        'Tag','Axes1Text1', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941 0.752941 0.752941], ...
        'FontSize',14, ...
        'ListboxTop',0, ...
        'Position',[247.5 445.5 162 17.25], ...
        'String','Initialization Window', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'Callback',mat5, ...
        'ListboxTop',0, ...
        'Position',[12 402.75 51 15], ...
        'String','Load Shape', ...
        'Tag','Pushbutton3');
if nargout > 0, fig = h0; end
```