Manual for Macrospin Simulation Tool

1. Introduction

This is a manual for macrospin simulation tool. The simulation is performed on CUDA and C++, while the data processing and visualization are done using MATLAB.

Background of the macrospin simulation and its applications are introduced in the second section. In the third section, seven operation modes are introduced. The parameters in the header.h file will be explained in the fourth section. Later in the fifth section, the procedures to compile and run the code are illustrated, including those for the data processing with MATLAB. Finally in the sixth section, examples of the visualization results are shown.

2. Background

As the Moore's law is near its end, spintronics is one of the promising approaches to solve this problem. The logic state of a spintronics device is represented by the magnetization direction of a ferromagnetic layer (FM). If there is another nonmagnetic layer (NM) in contact with the FM and a charge current (Jc) is injected into NM, a spin current (Js) with the spin polarization σ will be generated, as illustrated in Fig. 1.

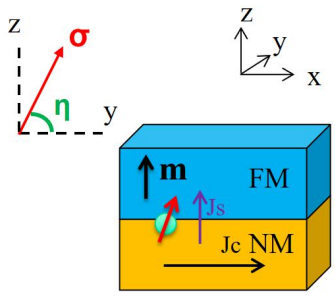

Fig. 1. Illustration of a spintronic device containing FM and NM layers. Spin current (Js) is generated when a charge current (Jc) is injected in NM. The spin polarization (σ) is in yz plane, associated with an angle η.

The spin polarization σ is in yz plane and is associated with an angle η. Such spins will exert a torque

on the magnetization (m) of FM and change the direction of m. The evolution of m can be calculated by Landau – Lifshitz – Gilbert (LLG) equation [1]:

$$\frac{d\hat{\mathbf{m}}}{dt} = -\gamma\hat{\mathbf{m}} \times \mathbf{H}_{\text{eff}} + \alpha\hat{\mathbf{m}} \times \frac{d\hat{\mathbf{m}}}{dt} + \gamma c_{j,D}\hat{\mathbf{m}} \times (\hat{\mathbf{m}} \times \hat{\boldsymbol{\sigma}}) + \gamma c_{j,F}\hat{\mathbf{m}} \times \hat{\boldsymbol{\sigma}}$$

Therefore, if the charge current is large enough, m will be switched from up to down, which also changes the information stored in the device.

In this macrospin simulation tool, parallel computing is used to simulate the evolution of m caused by the spins. Parallelism is natural for this task. In order to find the switching current, a sufficiently large range of Jc should be simulated. In addition, there are also many other parameters that can be changed (e.g., different situations when considering thermal effects). Such a large number of different cases are independent, so they can be computed in parallel. By using parallel computing, the performance can be greatly improved, enabling us to do computationally expensive simulations that CPU cannot achieve in reasonable time.

3. Operation Modes

The code contains both CPU and GPU versions. For GPU version, two iteration methods can be chosen, which is Euler method (argument 1) and Heun method (argument 2, 5, 6 and 7). To obtain a phase diagram of final mz with respect to injecting currents J and eta angles, multiple cases argument should be used (argument 1, 2 or 7), which will output the final mz at the last iteration step for these cases. To get the trajectory of magnetization during the simulated time, one case argument (argument 5) should be used and the mx, my and mz during the whole simulation process will be the output. GPU version also supports single eta mode (argument 6), which will output the final mz for different current values but one specific eta value. All GPU modes consider the rise/fall time (rfT) except argument 2. CPU version only has two types and both use Euler method. One calculates multiple cases on CPU (argument 3). The other one is one-case calculation, which will output mx, my, mz for all the iteration steps for one case (argument 4). The CPU version does not consider rfT. The summary of operation modes and output files for the seven arguments are shown in Table 1.

Argument Value	Operation Mode	Output
1	GPU, Euler w/ rfT	mz.txt
2	GPU, Heun (stochastic) w/o rfT	mz.txt
3	CPU, Euler w/o rfT	CMz.txt
4	CPU, Euler, one case w/o rfT	m_x.txt, m_y.txt, m_z.txt
5	GPU, Heun, one case w/ rfT	m_x.txt, m_y.txt, m_z.txt
6	GPU, Heun, single eta w/ rfT	mz_eta0.txt
7	GPU, Heun w/ rfT	mz.txt

Table 1. Argument values and the corresponding operation modes and output.

4. Input Parameters

In the header.h file, there are plenty of input parameters that can be tuned, which are summarized in Table 2, together with their default values.

Name	**Meaning**	**Unit**	**Default Value**
damping	damping constant		0.005
Msat	saturation magnetization	emu/cm^3	1000
anisotropyField	anisotropy field Hk	Oe	4000
tFM	thickness of FM layer	cm	1E-7
tp	time of the current pulse	s	200E-9
ttot	total simulation time	s	240E-9
deltat	time interval for all modes except CPU one case	s	1E-12
deltat2	time interval for CPU one case	s	1E-12
DL	damping-like torque efficiency		0.3
FL	field-like torque efficiency		0
T	temperature	K	0
Area_cross_section	area of the cross section of device	cm^2	900E-14

thermal_step	situation number for thermal effect study		1
eta_start	starting point of eta value and also single eta mode	°	0.1
eta_end	ending point of eta value	°	90
eta_step	number of eta points		100
J_start	starting point of J value	A/cm^2	8E5
J_end	ending point of J value	A/cm^2	3E8
J_step	number of J points		1000
Hx	applied external magnetic field along x	Oe	0
rise_time	rise time of the current	s	0.2E-9
fall_time	fall time of the current	s	0.2E-9
eta_fixed	eta value for one case simulation	°	60
J_fixed	J value for one case simulation	A/cm^2	3E7
BLOCK_SIZE	size of each block		1024

Table 2. Input parameters and the corresponding default values

Recommended parameters

***If thermal effect is not considered, thermal_step should be set as 1 and T as 0.**

The following parameters are changed based on default values. Run time is calculated using Tesla T4 GPU.

1. Heun method (argument 2 or 7) is the best mode. The default parameter values are suitable for Heun method. (Runtime ~= 1 s for argument 2 and 1.6 s for argument 7)

2. For trajectory, argument 5 using GPU and Heun method is recommended. (Runtime ~= 0.1 s)

3. If Euler method is used (argument 1), the results using the default deltat (= 1E-12 s) are not accurate. For more accurate results, change deltat to 1E-14. (Runtime ~= 150 s)

4. Considering the thermal effect requires changing thermal_step to, for example, 100. Change T to 300 for room temperature. Tune eta_step to 10 to save run time. Note that deltat should be 1E-12 and Heun method (argument 2) should be used. (Runtime ~= 27 s)

5. To calculate mz for single eta value, change the eta_start and Hx as needed, use argument 6. (Runtime ~= 1.7 s)

6. To use CPU multiple case mode (argument 3), change tp to 2 and ttot to 4. Change eta_step to 10. For accurate results, change deltat to 1E-14. (Runtime ~= 7000 s using deltat = 1E-14)

7. For trajectory and computation on CPU, use the argument 4 and change tp to 2 and ttot to 4 to save time. For accurate results, change deltat2 to 1E-14. (Runtime ~= 2 s)

5. Procedures to Run the Code

5.1 Simulation on CUDA Machine

Step 1. Load the files (header.h, helper_math.h, kernel.cu, main.cu, Makefile, macrospin_gold.cpp) to GPU server that supports CUDA.

Step 2. Change the gencode in Makefile according to the GPU version. e.g., change the gencode as NVCC_FLAGS = -I/usr/local/cuda/include -gencode=arch=compute_75,code=\"sm_75\" for a Tesla T4 GPU.

Step 3. Change the parameters listed in Table 2 in header.h as needed. You can also leave as it is and use default values.

Step 4. Compile the files by typing make.

Step 5. Choose an operation mode in Table 1 and type ./macrospin 2 for Heun method, for example. There is one argument that can be chosen between 1 and 7 and the argument is required.

5.2 Visualization

Step 1. Open plot_output.m using MATLAB.

Step 2. The simulation in 4.1 should output a mz.txt (CMz.txt) file for multiple case modes using GPU (CPU). Copy this file to the directory of plot_output.m. If the argument 3 is used, modify line 4 from mz.txt to CMz.txt.

Step 3. Modify the parameters between line 15 and 27 and keep them consistent with those used in the simulation. If default values are used in the simulation, no modification is needed.

Step 4. Run the code and the phase diagram and Jsw-eta plot will show up.

Step 5. This step is only required if you would like to plot the trajectory and use the one-case GPU/CPU mode (argument 5/4). Copy m_x, m_y and m_z.txt to the current directory. Uncomment the last block between line 96 and 118. Run the code and you should see the trajectory of the magnetization.

Step 6. This step only applies to single eta mode (argument 6). If the single eta value is 0, copy the mz_eta0.txt to the current directory and change the variable contain_zero to 1. After running the code, switching current J0 is calculated.

6. Examples

6.1 Default parameters, Heun method for phase diagram (argument 7)

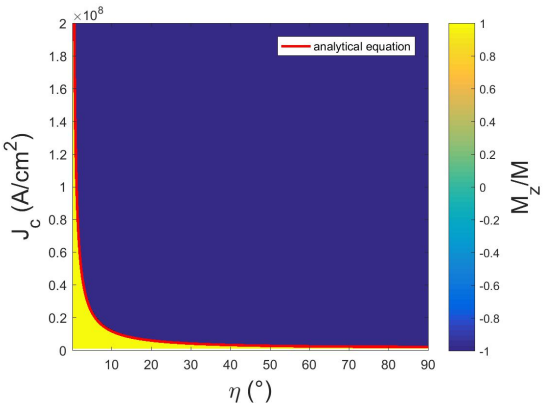

Fig. 2. Final magnetization phase diagram using default parameters. Critical switching current decreases with increasing eta angle. Simulated results agree well with analytical equation.

6.2 Default parameters, Heun method for trajectory (argument 5)

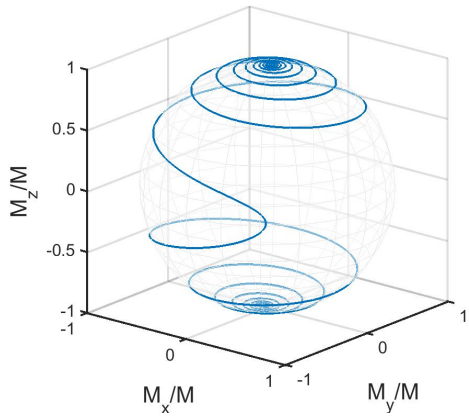

Fig. 3. Magnetization trajectory using default parameters. Magnetization is switched from up to down.

6.3 Parameter changes: eta_step = 10; T = 300; thermal_step = 100, Heun method for stochastic switching (argument 2)

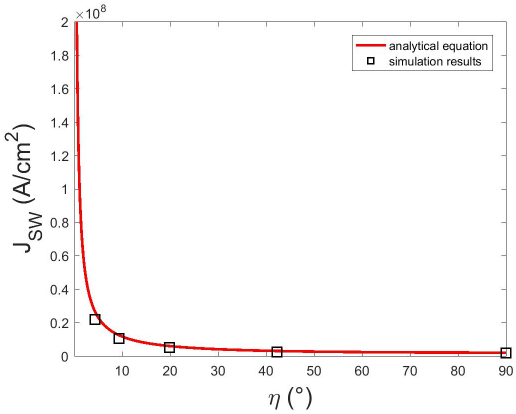

Fig. 4. Switching current density versus eta values at room temperature. For eta values that are smaller than 4.38°, non-deterministic switching occurs, namely, the magnetization will randomly point up or down.

References

[1] Lee, D.-K. & Lee, K.-J. Scientific Reports 10, (2020).

[2] CUDA Samples library for float3 operations in kernels.
https://github.com/zchee/cuda-sample/blob/master/common/inc/helper_functions.h