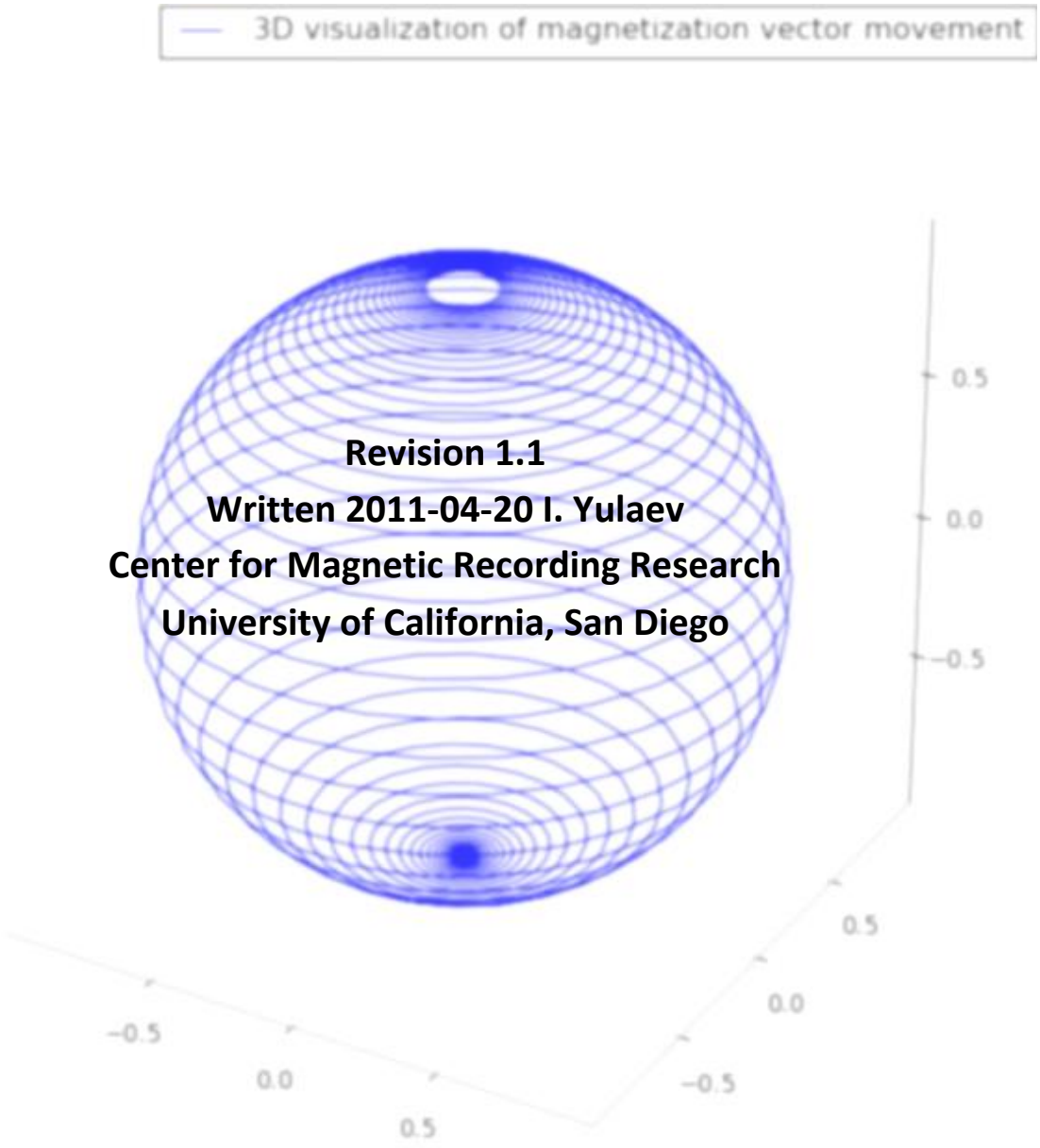
# Spinsim GSL Simulator Documentation




**Revision 1.1**

**Written 2011-04-20 I. Yulaev**
**Center for Magnetic Recording Research**
**University of California, San Diego**

# 0 Table of Contents

## 0.1 Table of Figures



## 0.2 Table of Equations

# 1 Overview

Spinsim GSL is a macromagnetic simulator of multi-layered nano-magnetic thin film structures. It is geared towards simulation of the behavior of thin film magnetization under stimulus from perpendicularly-applied currents and fields. It is particularly suited for simulation of magnetization dynamics due to spin torque transfer, as modeled by the Landau Lifshitz Gilbert equations with the Slonczewski Spin Torque Transfer term. One or two free layers, plus an optional fixed layer, can be simulated; inter-layer interaction in the form of mutual demagnetizing field and exchange coupling can be simulated. A GUI is provided for setting up all of the relevant parameters for the model. Plotting of results is supported.

This program was developed chiefly by Ivan Yulaev, a former undergraduate student at UCSD, working under the direction of Professor Eric Fullerton of the UCSD Center for Magnetic Recording Research. Stephanie Moyerman assisted greatly with the underlying mathematics, as well as writing the initial drafts of the code.

## 1.1 Licensing Terms



## 1.2 Change Log

This section describes the change history for this manual.

**Revision 0.1 – 2010-03-13**
Initial revision

**Revision 0.2 – 2010-12-24**
Revision to add numerous added features.

**Revision 0.3 – 2011-01-03**
Updated chapter 3,4 to latest GUI options
Updated code overview diagrams in chapter 7
Added chapter 5, “Compile Time Features”
Added chapter 6, “Errata”

**Revision 0.4 – 2010-01-06**
Updated section 1.5 to add note on OS dependencies.
Updated sections 1.7 – 1.9.

**Revision 0.5 – 2010-01-13**
Added libpng to sections 1.5.2 and 1.5.3.
Updated section 5.9 - FMR Resonance Simulation

**Revision 1.0 – 2011-04-07**
Added sections 5.11 - Removing Mutual Dipole Field and 5.12 - Swapping H and/or $J_{current}$ with J
Added section 5.5.1
Added sections 3.5, 3.2.2
Many small changes and revisions throughout
Manual is now considered "production ready"

**Revision 1.1 – 2011-04-20**
Added acknowledgements (section 1.3)

## 1.3 Acknowledgements

The Spinsim GSL project would not have been possible without the contributions of Marko Lubarda (under Prof. Lomakin and Prof. Fullerton, UCSD). His advice was instrumental in tightening up the simulation results. Christel Berthelot and Stephane Mangin (Nancy) contributed greatly to the early revisions of the code. Spinsim GSL would certainly not have been possible without the work and guidance of Stephanie Moyerman. Her advice and efforts were instrumental in transforming the simulator from a simple Matlab script to the program that it is now.

## 1.4 Terminology

Some terms in this document are used interchangably, and thus require disambiguation.

The terms "Spinsim" and "Spinsim GSL" both reference the same project, namely, this simulator project.

"Iteration" is used to refer to a time-independent current and applied field value pair, with which the magnetization of the structure is calculated. Typically, the spinsim simulator output will consist of the final magnetization of a layer for a grid of applied current/applied field pairs. Each point on the grid is an "iteration".

## 1.5 Simulation Accuracy

Spinsim GSL was developed to allow for very rapid simulation of magnetization behavior and thus efficient exploration of parameter space for magnetic thin films. It sacrifices accuracy in exchange for very fast run times. Commercial micromagnetic simulations can often provide greater accuracy of results at the expense of long run times. Part of the Spinsim GSL development process was correlation of simulation results to "known good" micromagnetic simulators. More specifically, for numerous common models, results from the Spinsim GSL simulator, particularly the measured critical switching current for a free layer, were compared to commercial micromagnetic simulation results. It was found that for most of the models, the Spinsim GSL simulations determined $I_C$ to within 10% of the value measured by a micromagnetic simulator.

As for all simulation tools, it is important to run many simulations on known models before using the simulator to explore new areas. Simulation tools cannot take into account real-world imperfections in

magnetic materials, which can cause simulation results to deviate significantly from those measured empirically. *The creators of Spinsim GSL neither claim nor imply any warranties regarding the fitness or performance on the Spinsim GSL simulator. The simulator is provided "as is".*

## 1.6 Dependencies and Required Packages

### 1.6.1 Operating System

Spinsim GSL was developed and run under Linux version 2.6. Any modern Linux distribution such as Ubuntu or Debian should work fine. It is recommended to use a distribution supporting the Debian package system, to simplify dependency management.

Running Spinsim GSL under Windows is not officially supported. The Python-based GUI frontend will run just fine provided you have Python and the required packages installed. A Python distribution like Python (x,y) works great for this. Running the main C-based engine is also possible, if the GSL and pthreads dependencies can be satisfied. Compiling and running the C engine under Cygwin should be possible. However, this is not explicitly supported.

For running Spinsim GSL under Windows, a virtualization tool like Sun VirtualBox can be used to create a virtual "guest" instance of Linux running on top of Windows. This approach is recommended if you wish to run Spinsim GSL on a computer with Windows as its primary operating system.

### 1.6.2 Required Packages

Some packages are required to compile and run this program. The program, as a whole, has been tested and developed under Debian Linux x86-64. All of the packages required may be obtained via Debian's Synaptic Package Manager. The following is a partial list of required packages:

- gsl-bin
- python-qt4
- python-matplotlib
- python-qt4-dev (*)
- libgsl0-dev (*)
- libgsl0ldbl (*)
- pyqt4-dev-tools (*)
- qt4-designer (*)
- libpng12-dev(*)
- libpng12-0

(*) denotes required for development only

### 1.6.3 Other Used Programs

Apngasm, an animated PNG assembler, is distributed with Spinsim GSL. This program was written by Max Stepin. It is used to create time-domain magnetization animations. This program requires the libpng12 runtime libraries to run, and development libraries to compile.

## 1.7 Compiling and Running the Code

Once you've verified you have all of the necessary packages in the section Dependencies and Required Packages, the code may be compiled by running "make all" from the root of the spinsim_gsl folder. Various make options are available, see the comments in makefile.

To run the GUI, enter a terminal and type "python spinsim_gui.py". After some loading, the GUI should open and you should see a window as below

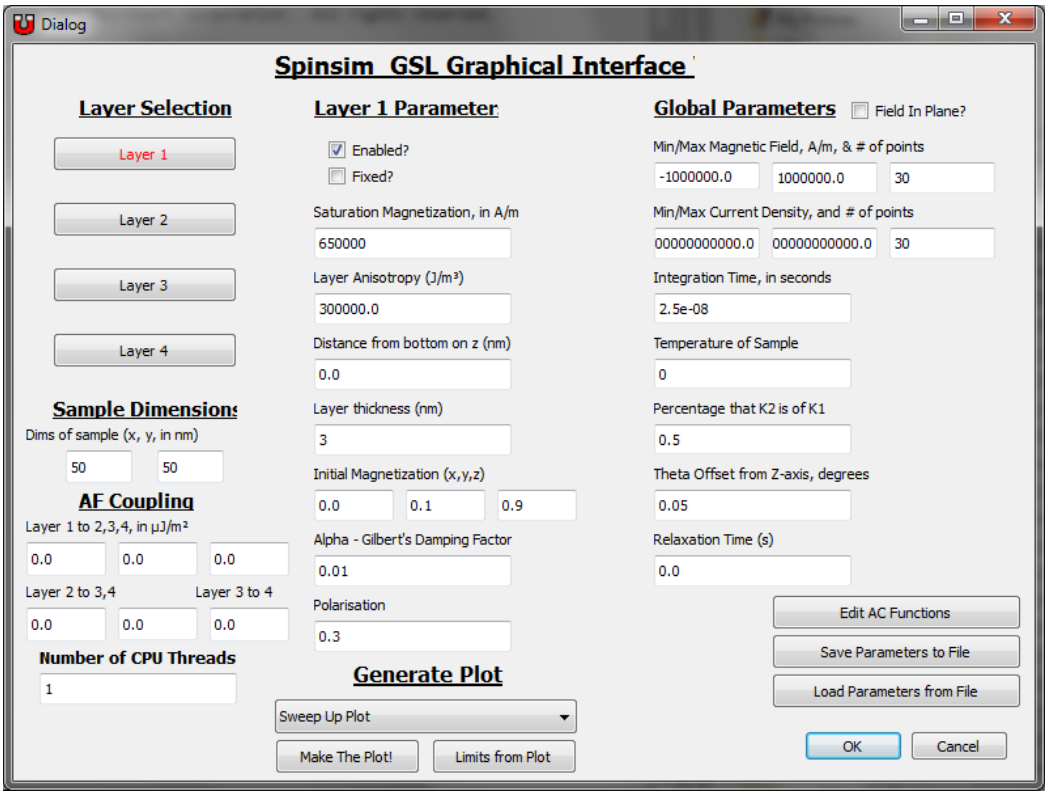



*Figure 1 - Spinsim GSL GUI Main Window*

Details for compiling and running may be found in the readme file, doc/README.

Instructions for using the GUI can be found in chapter 3.

For running the command line version, see chapter 4, "Spinsim GSL from the Command Line".

## 1.8 Programmers & Developers

Documentation of the code is outside of the scope of this manual. Most of the documentation may be gleaned from the comments in the C code and the docstrings in the Python code. Additionally, the code structure diagram (doc/code_structure_diagram.pdf) describes the organization of the C engine and Python GUI code, and the interfacing between these two. It is recommended that a developer review this diagram to better understand the structure of the program.

The code structure diagram can be found reproduced in section 7 of this document.

## 1.9 Useful References

The README file, located in the "doc/" sub-directory of the project, contains some instructions and technical details.

A tutorial is available under the "doc/" sub-directory of the project. It is recommended that new users of Spinsim GSL read the tutorial; it contains a walk-through of setting up and compiling Spinsim GSL, and using it to perform various simulations of simple magnetic elements.

The Landau-Lifshitz-Gilbert equation is the core of the computation done by the Spinsim GSL simulator. In essence, the function of the simulator is to set up the model as a magnetized layer and set of effective magnetic fields, and then model the behavior of the layer by integrating the LLG equation repeatedly. Wikipedia provides a brief overview of this equation.

Massimiliano d'Aquino's Thesis "Nonlinear Magnetization Dynamics in Thim-Films and Nanoparticles" provides a great introduction to the general topic Spinsim GSL operates within. His thesis may be found online at http://wpage.unina.it/mdaquino/PhD_thesis/main/main.html.

For a general discussion of spin torque transfer effects, Xiao, Zangwill, and Stiles provides a great overview in "Macrospin models of spin transfer dynamics" (Physical Rev. B **72**, 014446 (2005)).

Throughout this manual, other references regarding specific parts of Spinsim GSL are provided as relevant.

## 1.10 Contact

For questions or comments regarding the program, contact Ivan Yulaev (ivan_yulaev@yahoo.com).

# 2 Description of Model Used

## 2.1 Description of Structure Simulated

The below is a drawing representing the thin film nanomagnetic structure being simulated. It is an elliptical cylinder with 1-3 layers. Each layer has defined physical parameters such as anisotropy[1], saturation magnetization, alpha damping parameter, polarization, physical position and thickness, inter-layer exchange coupling, and whether the layer is fixed or free (i.e. whether or not the magnetization is allowed to change). Furthermore, some global parameters are specified: the overall dimensions of the model (dx, dy), the percentage that the 2nd order anisotropy term is of the first, and the offset of all layer's "easy axis" from the Z-axis along which field and current is applied.

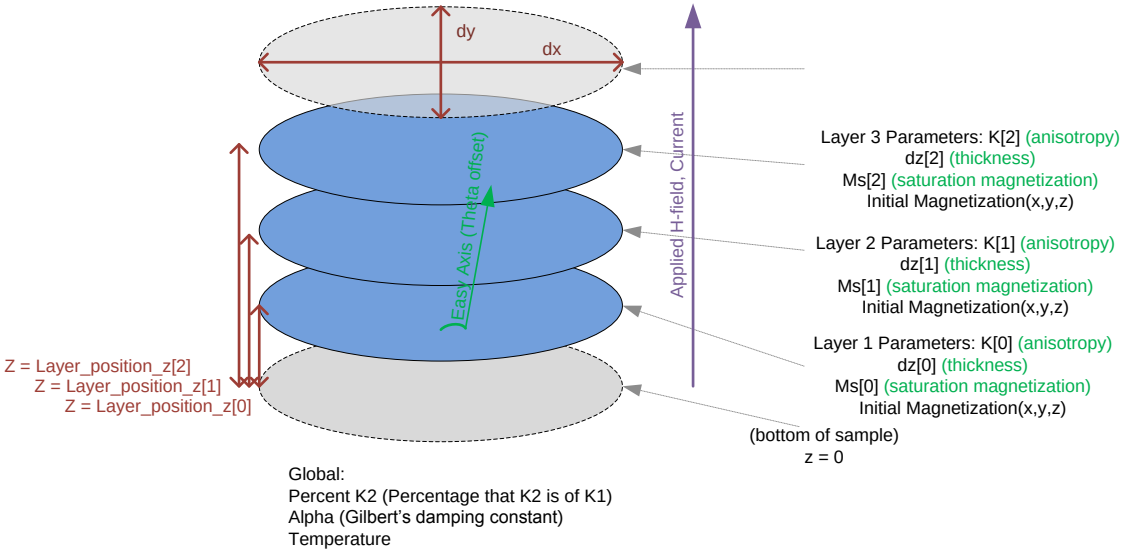



*Figure 2 - Drawing of Structure Being Simulated*

The simulator will calculate the static and dynamic behavior of the magnetization of the free layer(s) via integration of the Landau-Lifshitz-Gilbert equation. The parameters in the model are used to construct the "effective" magnetic fields they impart on each of the free layers to resolve the magnetostatic terms. An iterative solution of the LLG equation is performed to resolve the dynamic behavior resulting from model relaxation and the application of field and current. The simulator will report the magnetization of the structure after a given number of seconds has elapsed. Spin transfer torque is also

[1] Note that we assume it is uniaxial. We allow for a second-order $K_{U2}$ term, specified as a percentage of the 1st order term. This is discussed in more detail later in this chapter.

calculated for, although STT is not entered into the calculations as an effective field; it is evaluated using the spin transfer torque density term.[2]

## 2.2 Basis of Simulation

The Spinsim GSL simulator determines the magnetization state for each free layer by iteratively solving the LLG equation, with a spin torque transfer term.[3]

*Equation 1 - LLG Equation with STT Term*

$$\frac{d\hat{\mathbf{m}}}{dt} = -\gamma\hat{\mathbf{m}} \times [\mathbf{H}_{\text{eff}} + \mathbf{H}_{\text{T}}] + \alpha\hat{\mathbf{m}} \times \frac{d\hat{\mathbf{m}}}{dt} + \frac{\gamma}{\mu_0 M_s}\mathbf{N}$$

The bulk of the work done by Spinsim GSL goes into calculating the above terms from the parameters given and evolving the system in the time domain. The calculation of $\mathbf{H_{eff}}$ comprises what is known as the magnetostatics of the problem, and the calculation and evolution of the differential equation comprises the magnetodynamics.

## 2.3 Detail of Model Parameters

In this section, we will discuss the different parameters that may be specified to define the model. When possible, a description of how the parameter is used will be provided. All of the parameters may be specified through the GUI interface.

### 2.3.1 Units and Constants

In general, SI units are used throughout Spinsim GSL. A full discussion on the units and conversions can be found on Wikipedia[4]. Note that there is an option to use CGS units; see section 3.2.2 for the procedure to enable this.

Some constants are used explicitly in the Spinsim GSL simulator. These are

1. Boltzmann Constant ($k_B$) = 1.38 x $10^{23}$ J/K.
2. Gyromagnetic Ratio ($\gamma$) = 2.21 x $10^5$ m/sA.
3. Magnetic permeability of vacuum ($\mu_0$) = 4 x $\pi$ x $10^{-7}$ H/m.
4. Electron charge ($q_e$) = 1.61 x $10^{-19}$ C.
5. Reduced Planck constant ($\hbar$) = 1.054 x $10^{-34}$ J * s

### 2.3.2 Saturation Magnetization ($M_S$)

In general terms, the saturation magnetization of a material is the maximum strength of the demagnetizing (B) field. In Spinsim GSL, it is expressed in units of A/m. In the case of the thin film

[2] See Xiao, Zangwill, Stiles, "Macrospin models of spin transfer dynamics" (Physical Rev. B **72**, 014446 (2005)) for a more detailed discussion of STT evaluation in the macrospin context.

[3] See http://wpage.unina.it/mdaquino/PhD_thesis/main/node47.html for a more detailed discussion of the LLG equation with STT.

[4] "International System of Units", Wikipedia. http://en.wikipedia.org/wiki/International_System_of_Units

macrospin approximation, we use the saturation magnetization to scale the self and mutual demagnetization effective fields. In order to calculate the self (mutual) effective demagnetizing fields, we calculate a self (mutual) demagnetizing tensor using an approach laid out by Newell et al[5]. The effective demagnetizing field can then be calculated as

*Equation 2 - Calculation of Demagnetizing Field*

$$\boldsymbol{H_D} = M_S(\widehat{\boldsymbol{N_D}} \times \boldsymbol{m})$$

Where $M_S$ is the saturation magnetization, $\mathbf{N_D}$ is the demagnetization tensor, **m** is the unit vector representing magnetization, and $\mathbf{H_D}$ is the resulting demagnetizing effective field.

### 2.3.3 Layer Anisotropy

The (uniaxial) anisotropy of a layer determines the energy difference between states where the magnetization lies parallel to the easy axis and where the magnetization lies perpendicular to the easy axis. It is expressed in units of J/m$^3$. In Spinsim GSL, layer anisotropy is dealt with as an effective field with z and x components, where the "easy axis offset" is expressed in degrees away from the z-axis. The expression used for the x and z components of the effective field are

*Equation 3 - Calculation of $H_{ani}$ Effective Field due to Anisotropy*

$$H_{Ani}\hat{\imath} = \hat{\imath}\frac{2K_{U1}}{M_S\mu_0}(\boldsymbol{m}\cdot\hat{\imath})sin(\theta)$$

$$H_{Ani}\hat{k} = \hat{k}\left[\frac{2K_{U1}}{M_S\mu_0}(\boldsymbol{m}\cdot\hat{k})\cos(\theta) + \frac{4K_{U2}}{M_S\mu_0}(\boldsymbol{m}\cdot\hat{k})(\boldsymbol{m}\cdot\hat{\imath})^2(\boldsymbol{m}\cdot\hat{\jmath})^2\right]$$

Where $\hat{\imath}$, $\hat{\jmath}$, $\hat{k}$, are the unit vectors along the x, y and z axes respectively, $\theta$ is the angle between the z-axis and the easy axis, **m** is the free layer unit vector, $K_{U1}$ and $K_{U2}$ are the first and second-order uniaxial anisotropy components, $M_S$ is the saturation magnetization, and $\mu_0$ the permittivity.

[5] The tensor terms are calculated assuming the model cross-section can be described by rectangular blocks, which is not completely accurate as we are dealing with slices having circular cross-section. For a discussion on how the tensor terms are calculated, see *A Generalization of the Demagnetizing Tensor for Nonuniform Magnetization* by Newell et al.

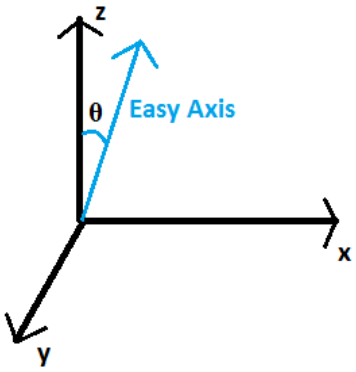



*Figure 3 - Illustration of Theta Offset function*

In Spinsim GSL, $K_{U1}$ is specified per layer. $K_{U2}$ is specified as a percentage of $K_{U1}$ and is specified for the system as a whole; it is numerically evaluated to an absolute value for each layer. The angle θ which determines the offset of the easy axis from the z-axis is also specified for the system as a whole.

### 2.3.4 Model Dimensions

Several dimensional constraints for the model can be specified. The model is assumed to be cylindrical, with the elliptic cross-section of the cylinder lying in the x-y plane. The x and y constraints must be specified globally, i.e. applying to all layers. The thickness and z position of each layer are specified per layer; the total thickness of the model is inferred from the thicknesses and positions of the layers.

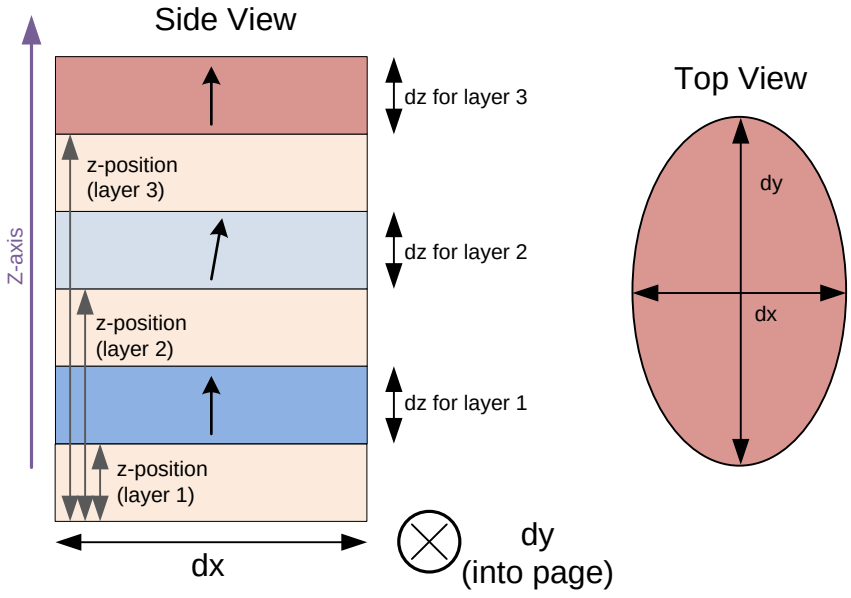



*Figure 4 - Illustration of Model Dimensional Constraints*

### 2.3.5 Initial Magnetization

The initial magnetization direction of each layer can be set by inputting an (x,y,z) vector in rectangular co-ordinates. This determines the starting magnetization orientation for each layer. For fixed layers, this magnetization will not change with time. The vector is normalized to unit length so it is not strictly necessary to ensure that the magnitude of the initial magnetization vector is exactly 1.

Note that on "spin up" runs, layer 0 will have its starting magnetization flipped across the z-axis, i.e. the z term **for layer 0 only** will be made negative. For "spin down" runs, the starting magnetization is used as input. This (ideally) simulates starting the free layer in different, but stable, states.[6]

### 2.3.6 Gilbert Damping Parameter (α)

The Gilbert damping parameter describes the damping of magnetization precession; it is used in calculation of the second term of the LLG equation. A higher α value results in greater damping and thus greater energy dissipation from magnetization motion. A parameters may be specified per layer. Note that the critical current to induce switching at very long timescales ($I_{C0}$) varies linearly with α; doubling α for a layer increases the layer's $I_{C0}$ by a factor of 2.[7]

### 2.3.7 Polarization

Polarization can be set per layer in Spinsim GSL. The polarization is used in computing the η(θ) (sometimes referred to as g(θ)) term in the spin torque transfer term from the LLG equation. The third term in the LLG equation can be expanded as below[8]

*Equation 4 - Expansion of Spin Torque Transfer Term from LLG Equation*

$$\mathbf{N}_{\mathrm{st}} = \eta(\theta)\frac{\hbar}{2e}\frac{J}{d}\hat{\mathbf{m}} \times [\hat{\mathbf{m}} \times \hat{\mathbf{M}}]$$

The η(θ) term is expanded as in Scheinfein's LLG simulator, using the formula[9]

*Equation 5 - Spin Torque Effiency Angular Dependence*

$$\eta(\theta) = \frac{4P_1P_2^{1/2}}{(1+P_1)^2(1+P_2)(3+cos\,\theta) - 16P_1P_2^{1/2}}$$

Plotting this function across angles from 0 to 180° we see

[6] Under some simulator settings, the layer 1 magnetization might be flipped across the z-axis instead. Or, both layer 0 and layer 1 may be flipped. See sections 5.3 and 5.4 of this manual for more information on how to enable this behavior.

[7] Tudosa, Katine, Mangin, Fullerton, "Perpendicular spin-torque switching with a synthetic antiferromagnetic reference layer."

[8] Xiao, Zangwill, Stiles, "Macrospin models of spin transfer dynamics" (Physical Rev. B **72**, 014446 (2005)).

[9] Michael R. Scheinfein and Elizabeth A Price, "LLG User Manual v2.50"

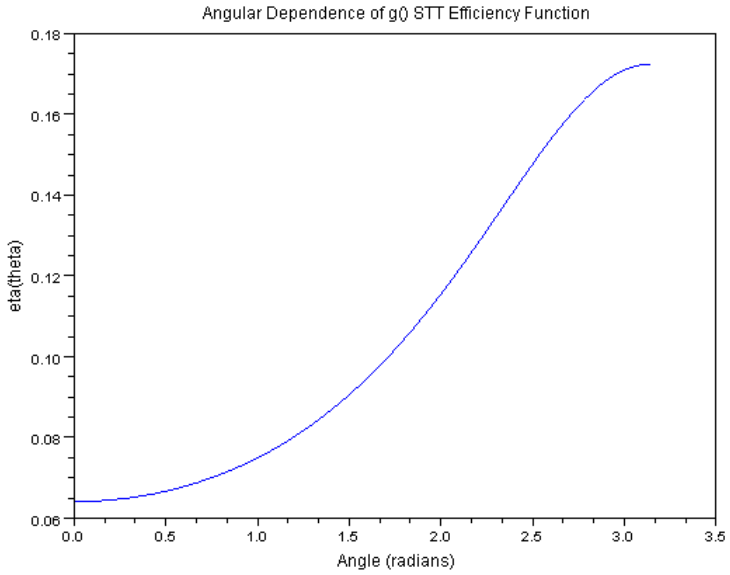



*Figure 5 - η(ϑ) Plot*

This, spin torque transfer efficiency is highest when the two layers through which current is driven are anti-parallel, and least efficient when they are parallel. Thus, for a structure with out-of-plane anisotropy, the current to switch from anti-parallel to parallel is generally lower than for the other direction.

### 2.3.8 Exchange Coupling

Exchange coupling is incorporated into the model. It is defined as a coupling constant ($J_{m,n}$) between layers, given in units of micro-Joules / $m^2$. A positive value for J translates into Ferromagnetic coupling, a negative value into anti-Ferromagnetic coupling. The coupling, given as an energy term, is converted to an effective field using the formula

*Equation 6 - Formula for Exchange Coupling Effective Field*

$$\boldsymbol{H_{exch}} = \frac{\boldsymbol{M} \times J_{m,n}}{10^6 \mu_0 M_{S\_n} dz_n}$$

for the exchange effective field from layer m onto layer n. In the formula, **M** is the unit vector representing magnetization direction of layer m, $J_{m,n}$ is the coupling between layer m and n, $\mu_0$ is the permittivity of free space, $M_{s_n}$is the saturation magnetization of layer n, and $dz_n$ is the thickness of layer n. This effective field is applied to layer n only; a similar expression is used to calculate the effective field for layer m.

## 2.4 Application of Field and Current

The simulator can be set up to apply a range of external field and/or current conditions to a model. These are applied along the z-axis; a positive field/current value corresponds to that field/current magnitude being applied in the positive z-direction.

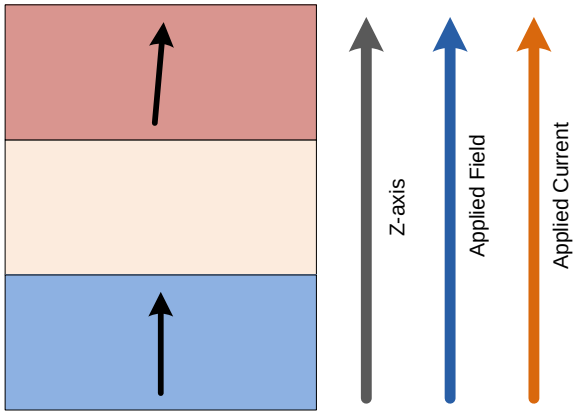



*Figure 6 - Illustration of Field/Current Applied Direction*

Typically, a range of fields and/or currents is specified, as well as a point density for the range. The simulator will run one simulation for each (field, current) pair in the ranges. For example, if the range densities for field and for current are both 20, 400 simulations in total will be performed. In general, a time-invariant field and current is assumed. Time-dependent functions can also be applied.

### 2.4.1 Relaxation Time

Before application of an external stimulus, it is often useful to allow a model to "relax" into a stable resting state, as it is often unrealistic to simulate a model starting in an unstable state. However, knowing exactly what a stable state is ahead of time may be difficult. A no-field/no-current relaxation of the model can be accomplished by defining a relaxation time in the Spinsim GSL interface. At the beginning of each simulation, the model will be put into its initial starting state defined by the initial magnetization vector, and then will be simulated for the duration of the relaxation time with no field nor current. At the end of the relaxation period, the external field and current condition for that simulation run will be applied and the simulation will continue as normal.

### 2.4.2 Time-Dependent Field and Current Functions

A time-dependent function can be used for describing the external field and current stimuli. The time-dependent functions can either be specified in terms of a sine wave (specify amplitude, frequency, and phase) or as a time-domain series of points, which will be linearly interpolated. In either case, the amplitude of the function can be specified absolutely, or can be specified as a scalar of each simulation run's time invariant field and current functions, for the time-dependent field and current amplitudes respectively. Any combination of time-variant/time-invariant functions may be evaluated and applied simultaneously via superimposition.

A discussion for specifying the time-dependent stimuli can be found in section 3.4. A description of the time-domain point series for user-defined time-dependent functions can be found in chapter 4.

# 3 Using the Graphical Interface

This section of the manual will describe the use and the features of the Spinsim GSL GUI. All functions will be explored, including setting up the model parameters, defining the stimuli, and plotting the results.

## 3.1 Launching the GUI

In order to run the program under Linux, several steps must be taken to ensure that the program can compile and all required libraries are present. Type "make all" on the command line. If there are errors *do not proceed further*; fix the problem, and attempt to compile again. Usually the issue is a missing package – the terminal should suggest what package is missing. All of the packages used by the tool can be acquired through package management software like Synaptic or aptitude – please see section 1.6 - Dependencies and Required Packages and make sure you have all of the packages listed therein.

Run the program. Type "python spinsim_gsl.py" on the command line. The GUI should come up as below:

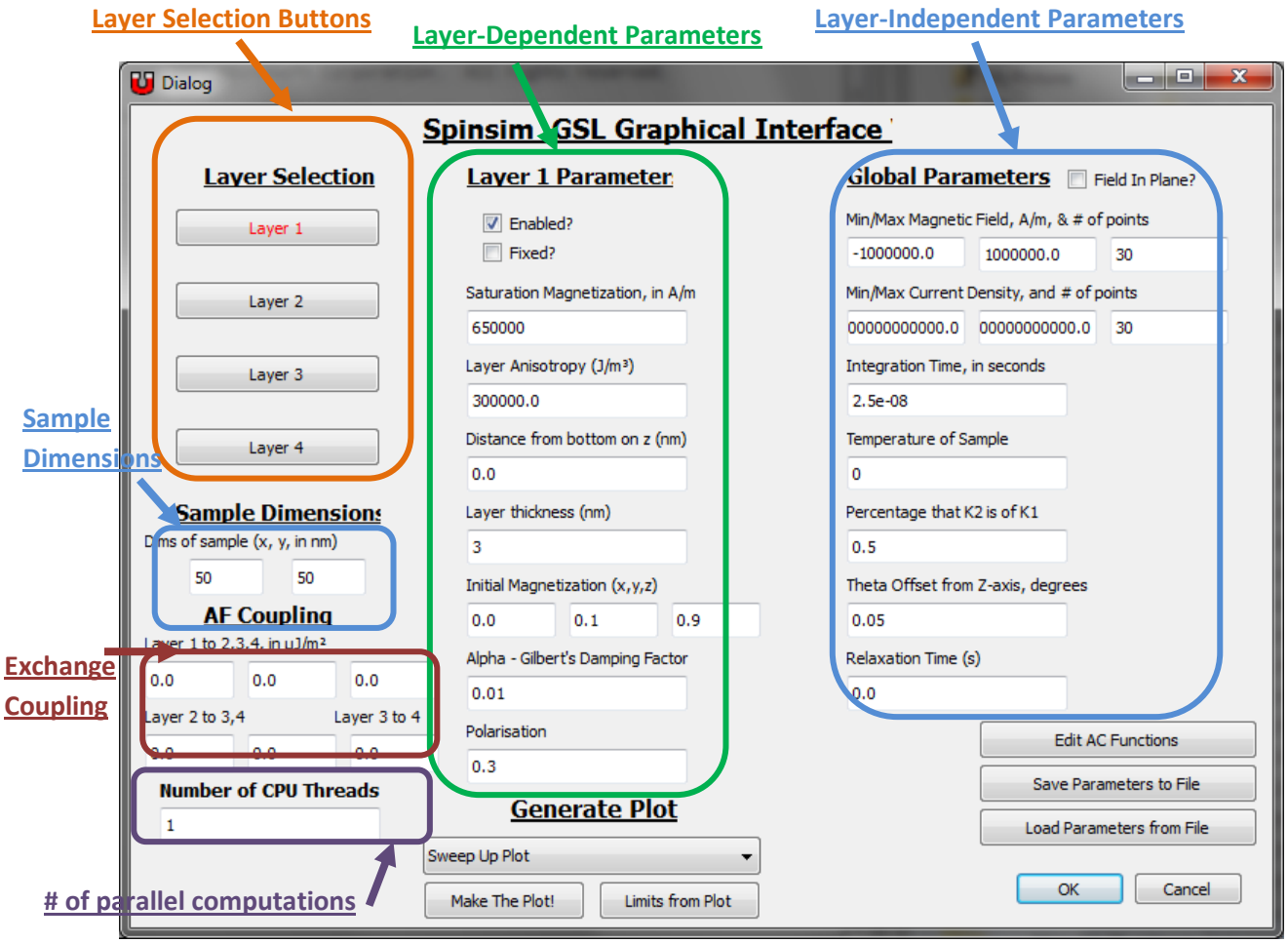



*Figure 7 - Spinsim GSL GUI Main Window*

## 3.2 Setting Up the Model Parameters

*For an explanation of the model parameters, please see section 2.3.*

There are two basic types of parameters that can be set in the GUI, layer dependent and layer independent (or global). The global parameters are always visible in the right-hand column of the GUI. These include simulator settings such as external field/current conditions, how many points to calculate along each axis, and how long to run each integration (i.e. simulation for one field/current value pair). Model settings include the temperature of the sample (for thermal noise)[10], the percentage that the 2nd order Anisotropy factor is of the first-order term[11], and the offset of the "easy axis" of the magnetic film relative to the Z-axis of the setup. Remember, external field and current are applied along the Z-axis.

The x and y sample dimensions are inexplicably located in the middle of the left-hand column. The z dimension of the model as a whole can't be set explicitly; it is implicitly determined by the vertical position and thickness of the top-most layer. Instead, vertical positions and thicknesses of each layer are entered. The vertical position specifies the distance from the bottom of the model to the bottom face of the layer.

Layer-dependent parameters can be set in the center column. These are set individually, per layer. The currently selected layer is colored **red** in the left-hand column. Each layer can be enabled and can have the magnetization fixed or "free", i.e. allowed the change as calculated by the LLG equation. Also, the saturation magnetization, layer anisotropy, initial magnetization, Gilbert's Damping Factor[12], and position and thickness of the layer can be set.

### 3.2.1 Exchange Coupling Constants

The exchange coupling values can be set in the bottom part of the left-hand column in the GUI. These constants are set for each pair of layers, and they are symmetrical, i.e. the exchange coupling between layers m and n is the same as between n and m.

A more detailed description of exchange coupling and how it is factored into the Spinsim GSL model may be found in section 2.3.8.

### 3.2.2 Setting the Units to Use

By default, Spinsim GSL operates in SI units. This unit system uses Ampere-turns per meter for magnetic field, and J for magneto-crystalline anisotropy energy. An alternative system of units is commonly used in the magnetics field; this is the CGS unit set, and it uses units of Oersteds (Oe) for H field, emu/cc for saturation magnetization, and erg/cc for magnetocrystalline anisotropy energy density. It is possible to switch to the CGS unit set in the Spinsim GUI by setting the variable cgs_units, defined at the top of

[10] Note that thermal noise is not supported, yet. See errata, section 6.1.
[11] Ralph Skomski, *Simple Models of Magnetism*, Section 3.1.3 "Higher-order anisotropies of nonuniaxial symmetry" (pg. 78)
[12] See "Gilbert's modification", "Micromagnetism" on Wikipedia, http://en.wikipedia.org/wiki/Micromagnetism. The α symbol in the Landau_Lifshitz-Gilbert equation is Gilbert's Damping Factor.

*spinsim_gui.py*, to be equal to 1 before running the GUI. Note that this will switch the units used in the GUI only; the commands passed to the actual spinsim engine will be passed as SI units. Furthermore, the plots will still inconveniently be in SI units.

## 3.3 Parallel Computation

Spinsim GSL is set up to take advantage of an arbitrary number of processors. In general, a given computation can be run on up to n processors, where n is the number of external current points to plot against. The number of threads to use can be set in the bottom-left corner of the GUI dialog.

Parallelization is done between different current values for simulations. Effectively, this means that simulations can only be parallelized for each current value; an H-field hysteresis loop consisting of only one current value but many H-field values can't be parallelized using the existing Spinsim GSL engine.

## 3.4 Time-Dependent Field/Current Function Setup

Aside from applying a time-invariant field and current for each simulation run, it is possible to apply a time-dependent field/current. Setting up the time-dependent field and current functions can be done in the "Edit AC Functions" dialog. Hit the "Edit AC Functions" button, in the bottom-right corner of the main dialog.

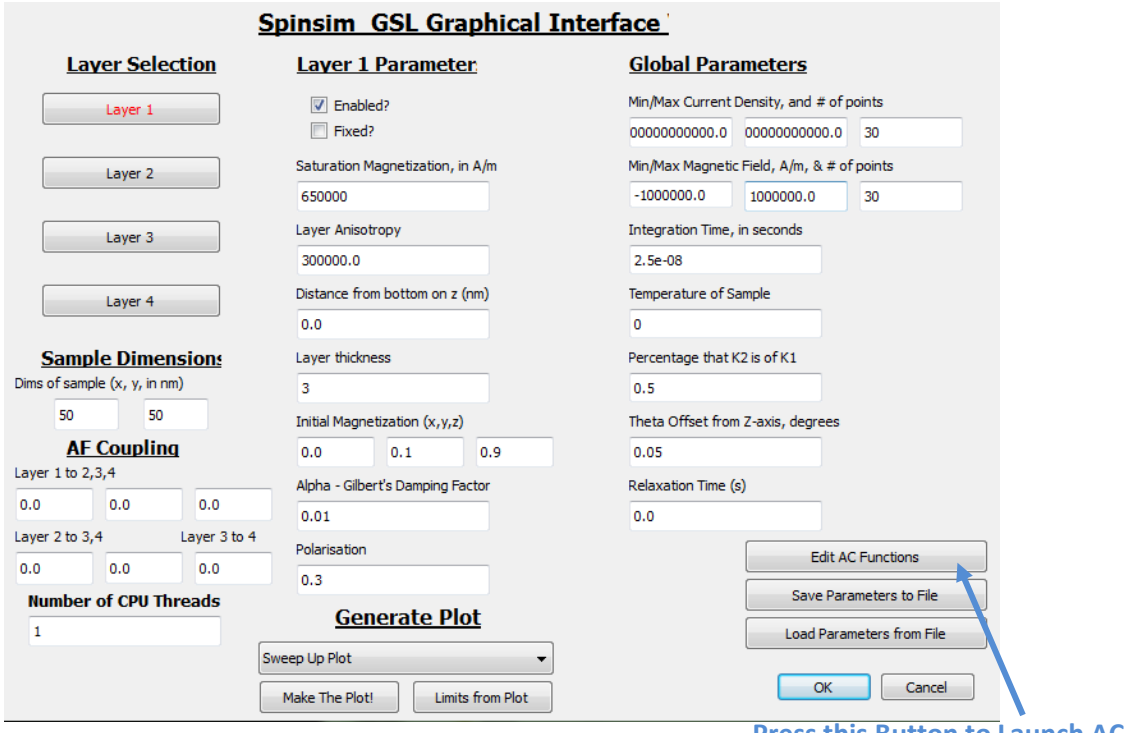



*Figure 8 - Edit AC Functions Button*

Once you press the button, the following dialog should appear.

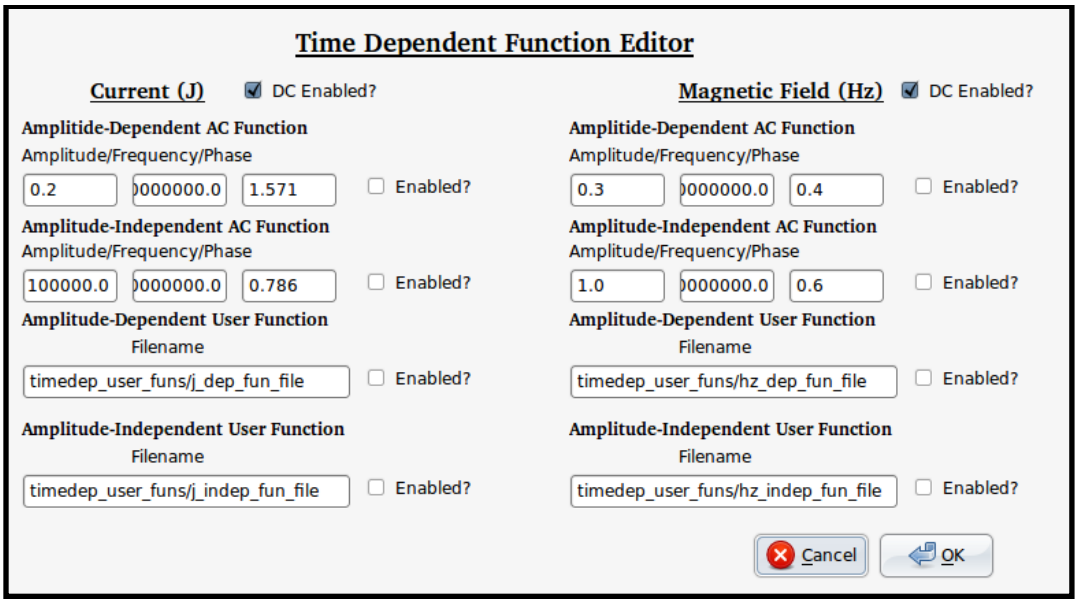



*Figure 9 - Time-Dependent Functions Dialog*

The function setup for external current and external magnetic field is identical. Current functions are set up in the left side of the dialog; applied magnetic field (H) is set up on the right side.

There are four types of functions that can be set up, for either applied field or current:

1. Amplitude-Dependent AC Function – This function generates a sine wave with the given frequency and phase. Frequency is given in Hz, phase in radians. The amplitude given is a scalar – the actual amplitude will be that scalar multiplied by the DC amplitude being used for a given iteration. So, for example, for a current AC function with an amplitude of 0.2, on the run where the DC current is 1000A/m$^2$, and amplitude of the AC sine wave will be 200A/m$^2$.[13]
2. Amplitude-Independent AC Function – Same as above, except that the amplitude is given absolutely, and will not vary from data point to data point. The amplitude is given in units A/m$^2$ for external current, and in A/m for externally-applied field.
3. Amplitude-Dependent User Function – The wave is defined as a series of (time, amplitude) points by the user in a tab-delimited file, with one point per line. The file path is given as the parameter for this function. An example file might be as below:

```
2e-9   0.1
20e-9  0.2
25e-9  0.3
```

   The first value on each line represent the time, and the second the "dependent" amplitude, which will scale from the current point's DC amplitude just as in the Amplitude-Dependent AC Function. Between time points, the integrator will use a linear approximation. For example, in the above case, at time 22.5e-9, it will use 0.25 as the amplitude scalar, as that is the midpoint between 0.2 and 0.3.

[13] In this example, magnitude will vary between -200A/m and +200A/m, for a peak-to-peak swing of 400A/m.

4. Amplitude-Independent User Function – The idea is the same as for the Amplitude-Dependent User Function, except the amplitude is specified absolutely. The amplitude is given in units $A/m^2$ for external current, and in A/m for externally-applied field.

For each type of function, the "Enabled?" box must be checked to use it in the simulation. All enabled functions are super-imposed upon each other. The settings are stored to the "fun_params" file in the main project folder; this file then gets loaded by the C engine and its contents used to set up the time-dependent functions in the C code.

## 3.5 Field Directionality

New in the 2011 version of Spinsim GSL, it is possible to simulate an in-plane field. This field will be applied along the x axis in the model. Current will still be applied along the z axis. To enable the in-plane field, check the "Field in Plane" dialog box, located in the top-right corner of the main Spinsim GSL dialog interface (Figure 7). The H field will now be applied along the x-axis.

Note that if FMR simulations are being run (section 5.9), this setting causes the bias (DC) field to be applied along the x direction and the AC field to be applied along the z-axis. Normally for FMR simulations, the bias DC field is applied along the z axis and the AC field is applied along the x axis.

## 3.6 Simulation Run Time

*See section 2.4 for a more detailed description of simulation time dependencies.*

Simulation run time can be specified in the right-hand column of the GUI, parameter titled "Integration Time, in seconds". This is the duration for which the behavior of the magnetic model will be simulated; after this amount of simulation time has elapsed, the magnetization will be recorded and output as the simulation result.

The relaxation time can also be specified in the right-hand column. Starting from t=0, until the relaxation time expires, no field or current will be applied to the model. This is described in section 2.4.1.

## 3.7 Running the Simulator

After setting up the layer dependent and independent variables, setting up the integration settings, the AF coupling constants, and the time-dependent current/field functions, you are ready to run the simulator! If you haven't set up the preceding, it's OK, the simulator comes with some reasonable default settings. Hit the "OK" button in the bottom-right corner of the dialog. A progress bar will appear in the bottom-left corner and start proceeding from 0 to 100. Once it gets to 100, the run is complete; you can view the data files manually in the project directory, or you can plot their contents to color plots from the GUI.

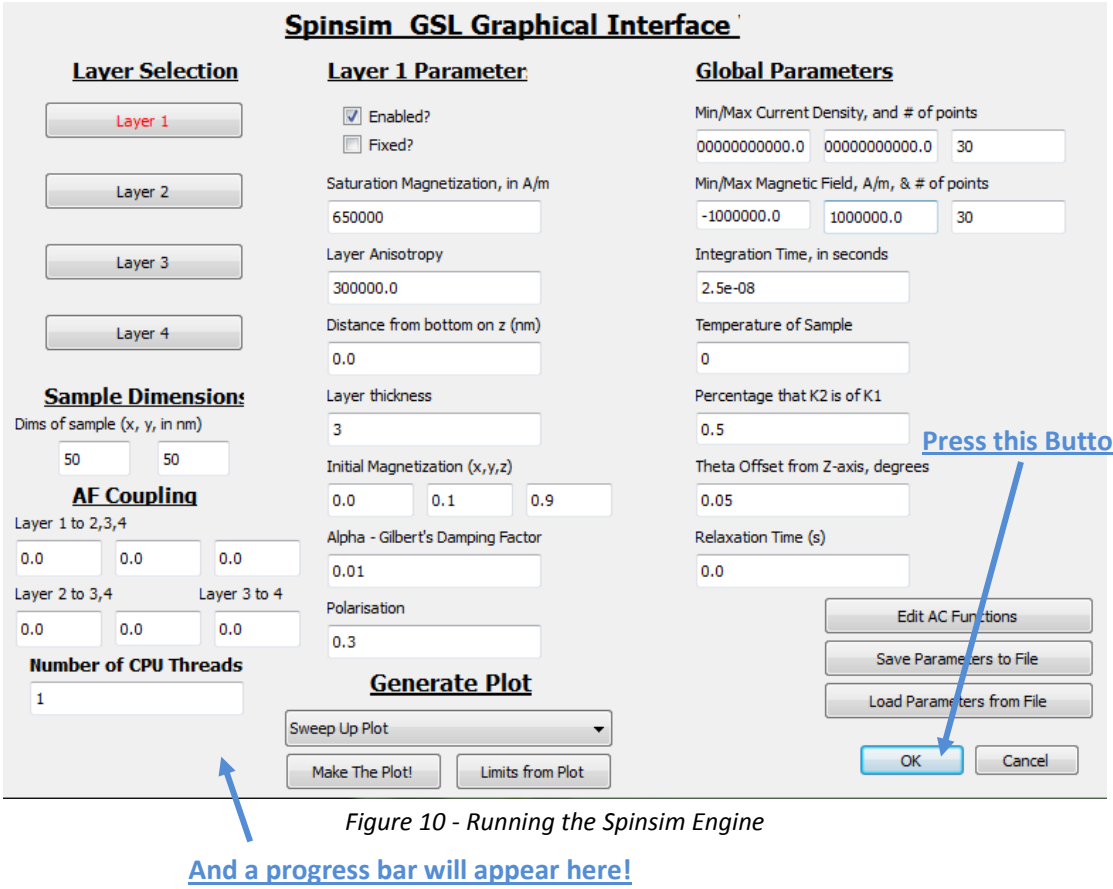



*Figure 10 - Running the Spinsim Engine*

The simulator (when run for field and current point density greater than 1) generates four files. These files contain the final magnetization for layer 0 (z-component only) as well as a description of the conditions used for each magnetization calculation.

1. magsweep_output_hz – a 1-dimensional list of external magnetic field conditions used in the plots.
2. magsweep_output-j – a 1-dimensional list of external current conditions used in the plots.
3. magsweepdown_output – the output from the down-ward sweep of magnetic field/current conditions. The top-left corner represent the minimum external field and minimum applied current; as you move right-ward, the field increases, and as you move downward, the current increases.
4. magsweepup_output – the output from the up-ward sweep of magnetic field/current conditions. The top-left corner represent the minimum external field and minimum applied current; as you move right-ward, the field increases, and as you move downward, the current increases.

The file layout is graphically depicted below:


*Figure 11 - Graphical Depiction of magsweep Output File Layout*

These files are written as tab-delimited tables of scientific ASCII representation of floating point values. Each value corresponds to the z component of the magnetization of layer 0, at the end of the simulation.[14]

See the example runs in the RUNS/ sub-directory to see example output from the Spinsim GSL simulator.

## 3.8 Plotting Results – Phase Plot

After running the simulator with a range of both field and current values, and generating the magsweepup/magsweepdown files, the plotting functions of the Spinsim GSL GUI may be used to generate plots for the magsweepup and magsweepdown data. There are 5 different phase plot types for the data:

1. Magsweepup plot – color plot of magsweepup data on applied field and current axes.
2. Magsweepdown plot - color plot of magsweepdown data on applied field and current axes.
3. Sum plot - color plot of magsweepup data summed with magsweepdown data on applied field and current axes.
4. Difference plot - color plot of magsweepup data minus the magsweepdown data on applied field and current axes.
5. All plots – the four above plots, all arranged in one figure.

An example of “all plots” is given below:

[14] The value output in the magsweepdown_output and magsweepup_output files can be changed; see section 5 of this document.

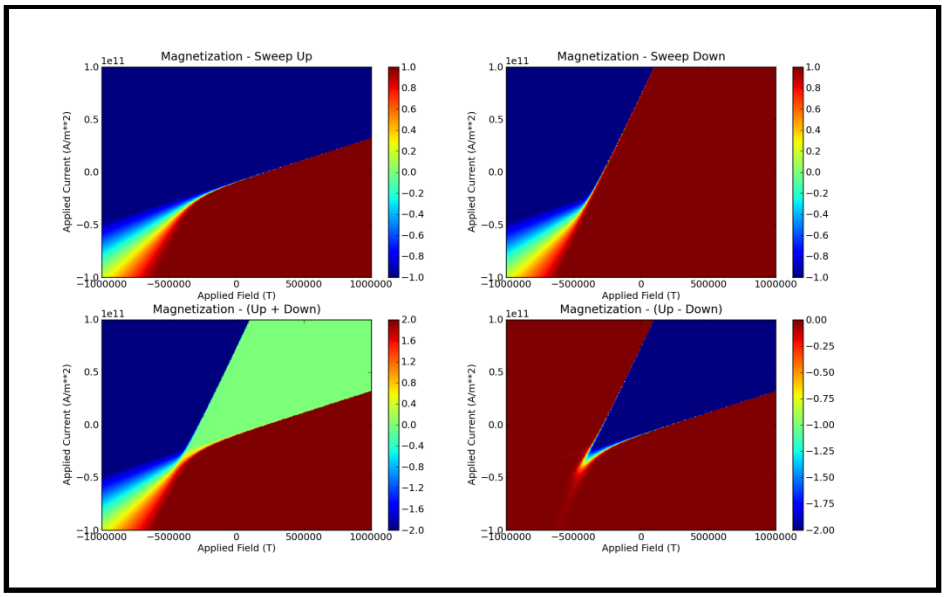



*Figure 12 – All Phase Plots in One Figure Example*

### 3.8.1 Extracting the Limits from the Plot

When viewing the plot window, it is possible to zoom and pan in the individual displays. If there is a particular region of interest you would like to re-plot with higher resolution, you can find that region by zooming in in the plot window, and then hit "Limits from Plot" in the main GUI. The minimum and maximum external field and current values in the Spinsim GUI main interface should change to represent the new, smaller window that you have selected.

## 3.9 Single-Point Simulation

When the field and current density fields (# of points to plot) are both set to 1, a special simulation case occurs. The simulation is run as before but instead of saving only the final magnetization at the end of the simulation, it saves the magnetization at every time step during the integration. The following files are created:

1. magsingle_x – list of x-axis magnetization for each time step, one point per line.
2. magsingle_y – list of y-axis magnetization for each time step, one point per line.
3. magsingle_z – list of z-axis magnetization for each time step, one point per line.
4. magsingle_t – list of the time value at each time step, one point per line

These results by themselves are not very interesting, but when used with the "Single Point" plot command, produce a figure as below:

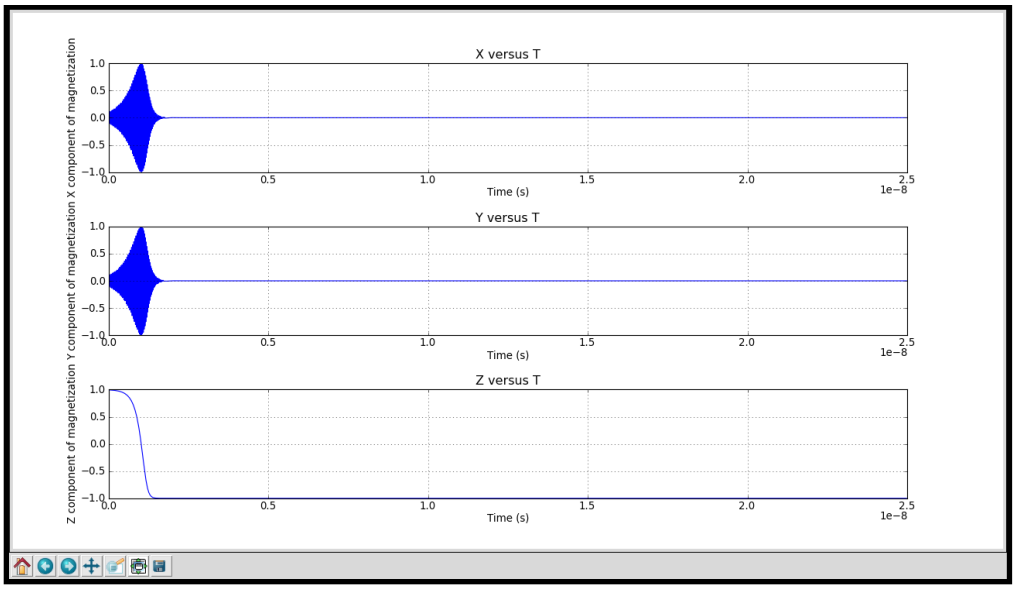



*Figure 13 - Single-Point Plot Example*

The x, y, and z axes are plotted individually versus time.

The “Single Point, Animation” plot option generated an animated PNG file of the change in magnetization. When this plot option is selected, a field for the number of frames appears. This determined how many frames will be generated for the animation. Once “Make the Plot” is hit, the frames are generated and assembled with apngasm. **Note that this will take a while, possible several minutes for 100 frames on a reasonably fast machine.** The animation will be saved as “magsingle_anim.png” in the main project folder.

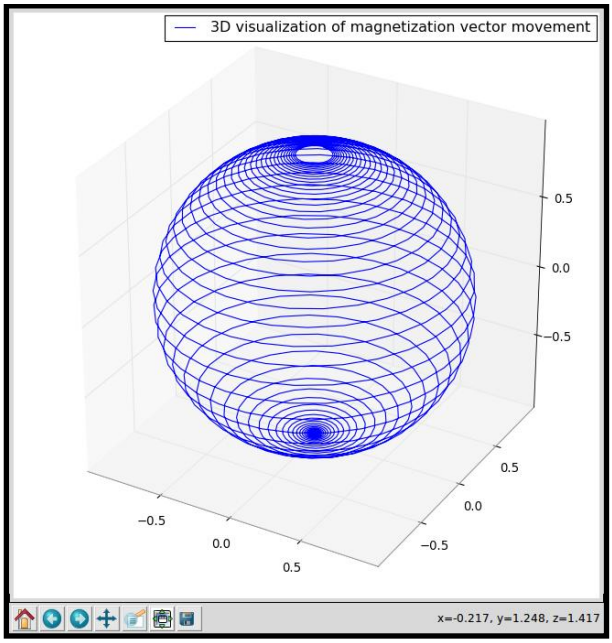



*Figure 14 - Example Frame from the Animated Single-Point Plot*

## 3.10 Hysteresis Loop Generation

When one and only one of either the current or field ranges has only one value, a hysteresis loop can be generated. When the # of points for current is set to 1 (while using a range of field values), choose “H sweep only”; for the opposite case where the number of field points is set to 1, choose “J sweep only”. A plot similar to the below will be generated.

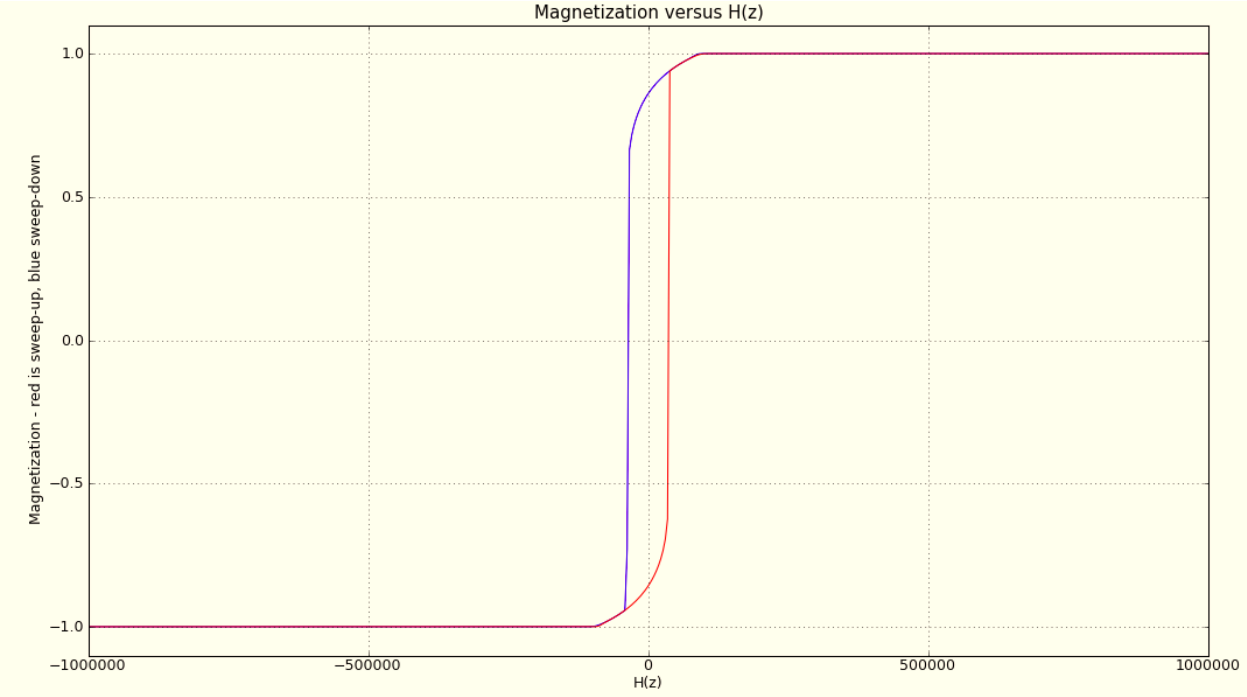



*Figure 15 - Example H-field Hysteresis Loop*

This function is useful for generating hysteresis loops and observing field/current-driven switching while holding other conditions constant.

The files created and their formats are the same as those described in section 3.7.

## 3.11 Saving & Loading Parameter Files

The Spinsim GSL has many parameters to specify. To ease the process of running particular simulations, it is possible to load and save parameters to human-readable INI files. This can be done by pressing the “Save Parameters to File” and “Load Parameters from File” button in the bottom-right corner of the GUI dialog. The buttons are indicated by the red arrows in the below dialog screenshot.

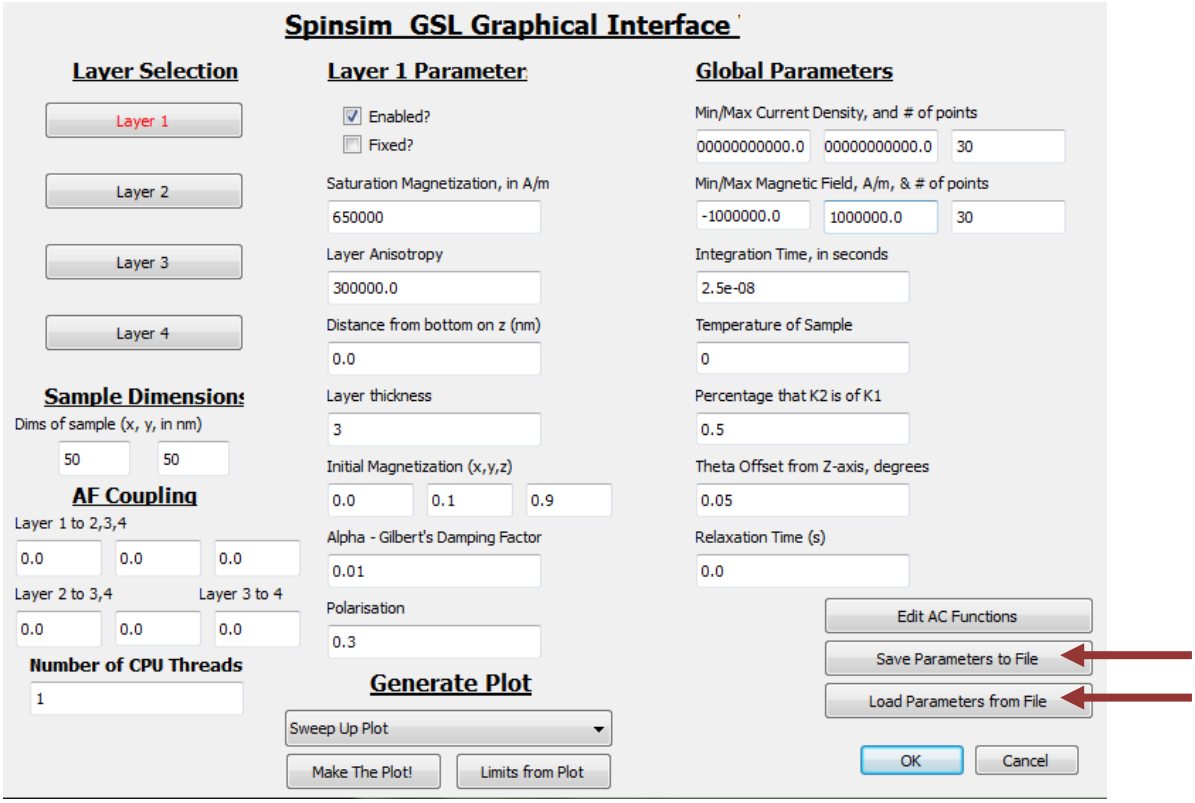



*Figure 16 - Save/Load Parameter Functions in GUI*

The files generated contain all parameters except the time-dependent function parameters – these are contained separately in the fun_params file. The files may either be generated by and re-loaded directly by the GUI or be edited by hand and loaded into the GUI, although the formatting is fairly restrictive and it is easy to make a mistake with the latter approach. Comments, which the simulator will ignore, may be added to the file by putting them on separate lines and pre-facing them with the '#' character.

# 4 Using the Command-Line Interface

The command-line interface for Spinsim GSL is more difficult to use than the GUI, primarily owing the fact that there are about 60 parameters that must be passed to the executable! However, the command line interface may be easier to use for scripted operation or for integration into a larger project. The use of the command-line interface will be described in this chapter.

## 4.1 Parameters

The following is a numbered list of parameters that must be passed to the GUI. These are to be passed as command line parameters, with spaces delimiting the values. Note that some parameters are floating points, while others are integers. For the executable, a floating point may either be written in "standard" form, i.e. *123.4567*, or "scientific" form, i.e. *1.234567e2*. Integers must be written in standard decimal form, i.e. *123*.

(1) The offset, in degrees, that the applied H-field is from the axis of the sample (floating point)
(2) The percentage that the 2nd order anisotropy factor K2 is of K1[15] (floating point)
(3) Gilbert's Damping Constant for layer 1 (floating point)
(4-7) Layers 1 through 4 saturation magnetization, in $A/m^2$ (floating point)
(8-11) Layer 1 through layer 4 anisotropy, $J/m^3$ (floating point)
(12-15) Distance of layer 1 through layer 4 from the bottom of the sample, in nanometers (floating point)
(16) Simulation type, least-significant 8 bits used. Each 2 bit pair encodes layers 1-4's types. Binary 00 means disabled, 01 means fixed, 10 means free. Layer 1 is the least significant 2 bits of this number (integer), the next layer is the next 2 more significant bits, etc
(17-19) Minimum external current density, maximum external current density, and number of external current density points to use (float/float/integer)
(20-22) Minimum external magnetic field, maximum external magnetic field, and number of external magnetic field points to use (float/float/integer)
(23) Integration time for the integrator, in seconds, Typically 5ns is long enough for a given run to settle into a polarized state but longer runs may be useful for observing oscillatory behavior (floating point)
(24) External current density time dependent functions, bit-wise OR'd. See section 4.3 for putting together these numbers for time-dependent current. For DC-only current, use "1". Otherwise, the fun_params file must be set up (integer)
(25) External magnetic field time dependent functions, bit-wise OR'd (integer)
(26-27) Sample size along x and y co-ordinates, in nm (floating point)
(28) Temperature for the sample, in K (floating point)
(29-40) Initial (x,y,z) magnetization vectors for layers 1-4 respectively. 3-tuple x,y,z for each layer should be given in that order. (floating point)

[15] Ralph Skomski, *Simple Models of Magnetism*, Section 3.1.3 "Higher-order anisotropies of nonuniaxial symmetry" (pg. 78)

(41-44) Thickness, in nm, for layers 1-4 (floating point)
(45-48) Polarization for each layer (floating point)
*(49) Sample shape (CURRENTLY UNUSED)* (integer)
(50) Number of computation threads to use. Shouldn't be greater than the number of current density points to plot, parameter 19. (integer)
(51-56) Anti-ferromagnetic coupling constant J between layers 1-2, 1-3, 1-4, 2-3, 2-4, 3-4 respectively (floating points)
(57-59) Gilbert's damping constant value for layers 2-4
(60) Relaxation time, in s, during which no field/current is applies to the model at the beginning of each simulation iteration.

Note if a layer is not enabled in parameter 16, its parameters must still be entered. But, the parameters will be "don't cares" so you can enter whatever you want.

Examples of command-line argument strings may be found in the readme file (doc/README).

## 4.2 Running the Simulator

First, be sure you've compiled the program by running "make" or "make all". Then, enter

./spinsim <parameter string>

with all parameters given a command line arguments, separated with spaces. The program will report its progress (in percent of total) as it is running. The output will consist of magsweep files just as described in section 3.5, or single-point files as described in 3.7 if parameters 19 and 22(field and current point counts) are both set to 1. These files may be plotted by the Spinsim GUI using the "Make Plots" button without having to re-run the simulator, or by calling spinsim_make_plots.py directly from the command line with the appropriate plot type passed as a parameter.

Some examples of valid simulator calls can be found in the README file (doc/README). Also, every time the GUI is run and the OK command is hit, it prints out the argument string used for its spinsim call, since the GUI is merely a front-end of the spinsim GSL executable. Both of these examples may serve as references for constructing valid parameter strings.

It is possible to set up the GUI to only output the argument string, and not actually run the Spinsim simulator. See section 4.5 for instructions on how to do this.

### 4.2.1 Making Plots from the Command Line

Simply enter

python spinsim_make_plots.py <plot type> <# of frames>

at the command line to create the plots. # of frames need only be given is the plot type selected is 7. The <plot type> parameter can be an integer representing one of the following

1. Magsweepup plot – color plot of magsweepup data on applied field and current axes.

2. Magsweepdown plot - color plot of magsweepdown data on applied field and current axes.
3. Sum plot - color plot of magsweepup data summed with magsweepdown data on applied field and current axes.
4. Difference plot - color plot of magsweepup data minus the magsweepdown data on applied field and current axes.
5. All plots – the four above plots, all arranged in one figure.
6. Single-point solution, plot individual Cartesian components vs time
7. Single point solution, make the animation
8. Fixed current, $H_{ext}$ sweep hysteresis loop.
9. Fixed $H_{ext}$, current sweep hysteresis loop.

The <# of frames> parameter parameter must only be given if <plot type> is 7. Note that, if a set of frames is generated with plot type argument 7, apngasm must be run to assemble the frames into an animated PNG. See the readme file for apngasm for instructions on how to do this, or look at the code in spinsim_make_plots.py that calls apngasm.

## 4.3 Manual Setup of Time-Dependent Current/Field Functions

In this section, we will describe how to enter time dependent field parameters. In the default case, the spinsim engine will use fixed applied field and applied current conditions for calculating magnetization for a given iteration. It is possible, however, to make applied field and applied current time-dependent so that, for example, it is possible to simulate the effect of passing an AC current through the structure.

There are two time-dependent waveform types defined for current and field – sine wave and user-defined points. These are both specified in external files and loaded by the spinsim program at runtime. Parameters are specified in a file named “fun_params”, located in the same directory as the spinsim executable.

### 4.3.1 Sine Wave Specification

Sine waves can be of two types, DC dependent or independent. In the dependent case, the amplitude of the sine wave is given as a scalar multiple of the time-independent waveform being used. In the independent case, the absolute amplitude is given. Both cases also require a frequency to be entered (in Hz) as well as a phase offset (in radians). Example syntax in the fun_params file is

j_sin_dep 0.2 1000000000 1.571

This line defines a dependent sine wave for current density. Its amplitude will be 20% of the magnitude of the current density used for the given iteration, its frequency will be 1GHz, and its phase offset will be PI/2.

j_sin_indep 1e5 1e9 0.785

This line defined an independent sine wave for current density. Its amplitude 10,000 A/m**2, its frequency will be 1GHz, and it's phase offset will be PI/4.

Similar syntax may be used for defining dependent and independent magnetic field density. The keyword at the beginning of the line should be “hz_sin_dep” and “hz_sin_indep” respectively.

### 4.3.2 User Waveform Specification

User waveforms may be specified using (time, value) pairs, to create arbitrary waveforms for a simulation. The values may be absolute current or field values, or may be scaling values for the current iteration's time-independent external field and current density values.

In the fun_params file, the user waveforms are specified using syntax like the following

hz_user_dep timedep_user_funs/hz_dep_fun_file

In this case, the user waveform for the external magnetic field will be loaded from the file timedep_user_funs/hz_dep_fun_file. The file is expected to consist of tab delimited time-value pairs, such as the following:

1e-9 0.1

10e-9 0.2

15e-9 0.3

Note that in the dependent case, the value is assumed to be a scaling factor for the time-independent current/field value. Other keywords for the fun_params file are hz_user_indep, j_user_dep, j_user_indep. These may be used to specify an independent field waveform, a dependent current waveform, and an independent current waveform, respectively.

### 4.3.3 Calling Spinsim from the Command Line with Time-Dependent Parameters

The 24$^{th}$ and 25$^{th}$ command-line arguments for the spinsim executable (see section 4.1) describe what time-dependent functions will be used. The values are treated as bitwise-ORed numbers, so that any combination of time-independent and dependent waveforms may be used for both current and field. The valid bits, starting from the least-significant bit, for these arguments are

0 – superimpose time-independent current/field value

1 – superimpose sine function that depends on time-independent current/field value

2 – superimpose sine function that doesn't depend on time-independent current/field value

3 – superimpose user function that depends on time-independent current/field value

4 – superimpose user function that doesn't depend on time-independent current/field value

These numbers may be added arbitrarily, to produce any combination of the above. For example, if one wished to use the time-independent current, a sine function that depends on the time-independent current, and a user function that is independent of the time-independent current, one would use as the command line argument

$2^4 + 2^1 + 2^0 = 19$

So, the 24th command line argument for calling the spinsim function would be 19.

Similarly, if one wished to use the same current setup as above but additionally, for applied field, use an absolutely-specified user function and the time-independent field value, one would use

$2^3 + 2^0 = 9$

as the 25th command line argument for spinsim executable.

### 4.3.4 Examples

For additional examples of specifying and using time-dependent current and field functions, please see the README file, the fun_params file included with the project, and the files in timedep_user_funs/.

## 4.4 Scripting Multiple Spinsim GSL Simulations

It is possible to script the command-line version of Spinsim GSL so that several simulations can be run in series without any user intervention. Creating a shell script is fairly straight-forward. Once the arguments for the spinsim command-line program are put together, it is only a matter of writing the .sh file. After each simulation, the simulator outputs should be moved to another folder, so that the next simulation does not over-write the previous one's output. This can be accomplished by writing the script as

```
#Simulation run #1
./spinsim <command line arguments for run #1 here>
#Make a new folder called simulation_1
Mkdir simulation_1
#Move the simulator output into a folder called simulation_1
mv mag* simulation_1/.

#Simulation run #2
./spinsim <command line arguments for run #2 here>
```

After writing such a script, the results, stored in separate folders, can later be plotted by copying the spinsim_make_plots.py file into each folder and generating the plots as described in section 4.2.1.

## 4.5 Using Spinsim GUI to Generate Command Line Argument Strings

Rather than putting together the argument strings that are given to the Spinsim GSL command line executable, it is possible to use the GUI to generate the strings, and then run them later (i.e. copy-pasting to the shell directly or using a shell script). To setup the GUI to NOT run the Spinsim GSL simulation and only print the command line string it would have used to run the simulator, go into the spinsim_gui.py file and set the variable *print_command_only* to 1. This will cause the GUI to output the argument string it *would have used* to run the command line simulator, but the GUI will not actually launch the simulator.

# 5 Compile-Time Defined Features

Some features in the Spinsim GSL simulator can only be specified at compile-time. This means that the make command must be run after making the required changes in the source code. These features are generally experimental, which may occasionally produce strange results, or are options that were only used for one particular simulation model. Nevertheless they may be useful for certain applications.

Most of the features are triggered by defines which can be commented/uncommented at the top of spinsim.c. There should already be an entry for each one (commented or un-commented) as well as a brief comment describing what the define does.

In C, a comment is any line beginning with the characters "//". In Python a comment is a line beginning with "#".

## 5.1 Calculation of GMR

By default, the spinsim simulator outputs, into the magsweepup_output and magsweepdown_output files, the z-axis magnetization of layer 0 for each (current, field) pair. If the define *CALCULATE_GMR* at the top of spinsim.c is uncommented, the output will instead be the GMR between the two layers, calculated by finding the cosine of the angle between the z components of layer 0 and layer 1 magnetizations.

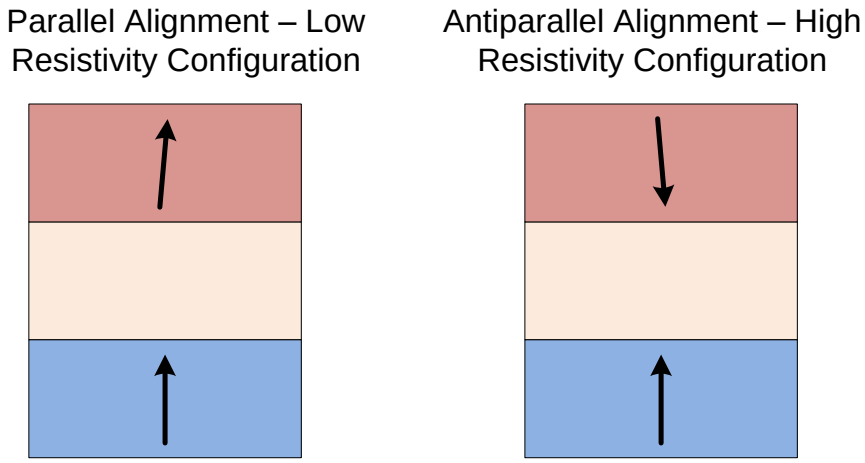



*Figure 17 - Schematic Illustration of GMR Effect*

## 5.2 Time-Averaged Magnetization

Spinsim GSL normally outputs only the final magnetization of layer 0. If the define *AVERAGE MAGNETIZATION* is uncommented then the time-averaged magnetization will be output.

If this is used in conjunction with *CALCULATE_GMR* (section 5.1) then the average GMR value will be output, as defined in the previous section.

## 5.3 Output Layer 1 Magnetization

Changing the value of the define *LOOK_AT_LAYER_2* to 1 will cause the simulator to output the final magnetization for layer 1 instead of layer 0. This is useful if both layer 0 and layer 1 are free. Note that also the magnetization of layer 1 will be flipped in the spin-up runs, instead of flipping that of layer 0.

If this feature is enabled, running the simulator with layer 1 specified as fixed may produce undefined output.

## 5.4 Switching Initial Magnetization of both Layers 0 and 1 for Spin-Up Simulation

Typically, the spin-up simulation switches the z-component of the magnetization for layer 0 (or possibly layer 1 if *LOOK_AT_LAYER_2* is defined as 1). If the define *SWITCH_BOTH_LAYERS* is uncommented the both layer 0 and layer 1 magnetizations are switched.

Note that this only works if both layers 0 and 1 are free. If layer 1 is fixed the magnetization of layer 1 will NOT be flipped.

## 5.5 Changing Polarization Model to Use

By default the symmetric Slonczewski form for polarization function (g(θ)) is used. This allows computation of g(θ) with only P specified for each layer. The asymmetric (AS) form may also be used by commenting out the define *POLARISATION_SLONCZEWSKI* and un-commenting *POLARISATION_XIAO*. In this case, the same function is used for each pair of layers for calculating g(θ) and $B_0$, $B_1$, $Q_+$ and $Q_-$ must be specified during compile time in the functions polarization() and polarisationpp(). These functions can be found in spin_helpers.c.

### 5.5.1 Modifying η(θ) Function Used

By default, the η(θ) function used to calculate STT in the Spinsim GSL simulator is the Symmetric Slonczewski Approximation. This function is given by Equation 5. This approximation, while being accurate for metallic junctions, is likely inappropriate for magnetic tunnel junctions, although contemporary research on this is not entirely conclusive. A more accurate model for MTJs, to the first order, approximates η(θ) as a constant. This mode can be enabled by un-commenting the CONSTANT_G_THETA define in the spin_helpers.c file.

Other η(θ) functions available are a symmetric function, which gives η'(θ) = η(θ) for θ = [0, π/2] and η'(θ) = η(180-θ) for θ = [π/2, π]. Uncomment the SYMMETRIC_G_THETA define in spin_helpers.c to enable this. A tunnel junction-like function, given by the below equation, may separately be enabled by uncommenting the define TUNNELING_G_THETA.

*Equation 7 - Spin Torque Effiency Angular Dependence, Tunnel Junction-like Form*

$$\eta(\theta) = \frac{P}{2(1 + P^2 \cos\theta)}$$

Note that for any of the changes above, the spinsim executable must be recompiled. Also please make certain to only enable, by uncommenting the appropriate define, one of the modified $\eta(\theta)$ forms.

## 5.6 True Sweep of Field/Current

Spinsim GSL typically re-sets the initial magnetization condition before every simulation run. To keep the magnetization state between runs so as to simulate "sweeping" field and current, comment out *RESTART_Y_FROM_INIT_VALS* and un-comment either *SWEEP_CURRENT*, to only re-set the magnetization to the initial values between different $H_{ext}$ values, or *SWEEP_FIELD_AND_CURRENT* to never reset magnetization back to the specified initial magnetization. Make certain that only one of these defines is uncommented at a time.

Note that the simulator sweeps through the field values for every current value given. That is, it does not proceed to the next current value until it was swept through all field values. For sweep-down, we start at the largest field value and sweep to the smallest (most negative). For sweep-up, it is from the smallest to largest field values.

Also note that parallelization is not reliable when using any function other than *RESTART_Y_FROM_INIT_VALS.* Set the number of threads to 1 if *RESTART_Y_FROM_INIT_VALS* is not defined.

## 5.7 Tracking Switching Points

If the define *TRACK_SWITCHING_POINT* is uncommented, the simulator will output the field/current values for which the magnetization is observed to switch, i.e. when the z-component of the magnetization settles to a value opposite that of the last run. The switching point results are output on standard out. This is useful f. ex. to calculate $H_C$ and $I_C$ in a model.

## 5.8 Track Settling Time

It is possible to track the settling time of a layer (settling after, for example, STT-induced switching) by uncommenting *TRACK_TSETTLING_Z* or *TRACK_TSETTLING_X*. If the former is uncommented, the settling time reported will be when the z component of the layer being observed (typically layer 0 unless *LOOK_AT_LAYER_2* is set to 1) is less than the value of the define *TSETTLING_CRITERIA_Z*. If the latter is uncommented, the settling time reported will be when the x component of the layer being observed is less than the value of the define *TSETTLING_CRITERIA*.

Note that settling time is inexplicably only tracked for sweep down runs. Also it may be somewhat inaccurate, as the amplitude of different magnetization components does not always reduce monotonically due to dipole field and coupling from other layers. It is always recommended to look at a single point trace to determine the true settling time of a layer, if the intention is to use the settling time as a proxy for damping.

## 5.9 FMR Resonance Simulation

Ferromagnetic resonance is a property of magnetic materials to absorb energy from an applied field if a field is of a particular frequency. If the define *FMR_FIELD* (found at the top of llg1.c and llg1_6.c) is uncommented, the simulator will apply only the DC component of the field along the z-axis, and will

apply the AC component along the x axis. This may be useful for doing experiments regarding FMR effects in magnetic materials and structures.

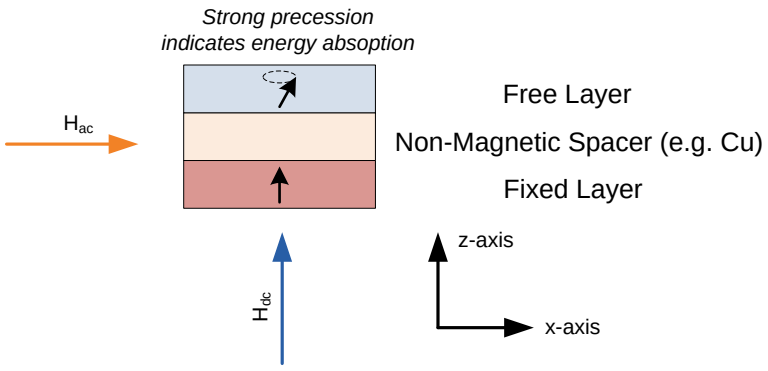



*Figure 18 - Diagram of FMR Resonance Simulation Setup*

As illustrated above, the DC component of the applied field $H_{dc}$ is applied along the z-axis, and the AC component $H_{ac}$ is applied along the x-axis. At the resonant frequency, the magnetic sample efficiently absorbs energy from the applied field and as a result precesses strongly. Such precession is indicative of a resonant condition.

The z-axis and x-axis applied fields may be swapped for in-plane FMR experiments. See section 3.5 for details on how to perform an in-plane simulation.

Note that this feature ONLY works for fixed-free and free-free model configurations, i.e. layer 0 must be free and layer 1 fixed or free. No other model configurations are supported.

## 5.10 Calculation of Exchange Energy

In a similar vein as the GMR calculation from section 5.1, it is possible to calculate an analog for the exchange energy and output that as the output of the simulator. If the define *CALC_EXCH_E* in spinsim.c is uncommented, the output of the simulator will be the sine of the angle between layer 0 and layer 1 magnetizations. This may be used as a proxy for exchange energy. A large value for the sine between the two angles should correspond to a large energy stored in the exchange coupling between the layers.

Unlike the GMR calculation, the exchange energy output takes into account all of the components of the magnetization vectors, not simply the z-component. The energy can generally be understood[16,17] as

*Equation 8 - Exchange Energy*

$$E_{exch\_m,n} = -J_{1,2}\sin(\theta_{1,2})$$

Since $J_{1,2}$ is constant, $\sin(\theta_{1,2})$ can be used as an analog for the exchange energy between the two layers.

[16] Note that here we implicitly assume bilinear coupling.
[17] See M.D. Stiles, "Exchange Coupling in Magnetic Multilayers" for more background.

## 5.11 Removing Mutual Dipole Field

It is sometimes interesting to remove the effects of mutual layer-to-layer demagnetizing fields when performing a simulation of a magnetic model. For some macrospin model setups, particularly the two layer fixed-free and free-free configurations, this can be done in Spinsim GSL. In the *llg1.c* and *llg1_6.c* files, uncomment the defined LLG1_NO_MUTDIPOLE and REMOVE_MUTUAL_DEMAG respectively.

Additionally, for the fixed-free configuration only, self-demagnetizing fields may be removed. To do this, uncomment the define LLG1_NO_SELFDIPOLE.

## 5.12 Swapping H and/or $J_{current}$ with $J_{1,2}$

For simulation of phase plots describing the effects of varying current, field, and exchange coupling, it may be useful to replace current or field with a constant value and sweep the exchange coupling between the two layers 0 and 1. This would create a phase plot with exchange coupling $J_{1,2}$ along one of the axes. For free-free configurations, both field/$J_{1,2}$ and current/$J_{1,2}$ swaps are possible; these can be enabled by un-commenting the defines LLG6_SWAP_JEXCH_AND_HK and LLG6_SWAP_JEXCH_AND_J respectively, located at the top of the file llg1_6.c. For fixed-free-free configurations, only field and $J_{1,2}$ may be swapped, and this can be enabled by uncommenting FXFRFR_SWAP_JEXCH_AND_HK at the top of llg1_fxfrfr.c.

In all cases, either the field or current (depending on what is being swapped) gets set to zero, and the min/max/step values for either the field or current sweep are used as the values for $J_{1,2}$. For example, if we are swapping field and $J_{1,2}$ in a two-layer free-free model, we uncomment the define LLG6_SWAP_JEXCH_AND_HK at the top of llg1_6.c and set field to sweep from 0 to 5000 A/m; the applied field H will be set to zero in the simulations and $J_{1,2}$ will be swept from 0 to 5000 μJ/m$^2$. In this example, applied current will be swept as usual, only applied field will be set to zero.

# 6 Errata

This section will describe errata. Errata is generally simulator behavior that deviates from what might reasonably be expected, or features that are demonstrated and documented but not fully or correctly implemented.

## 6.1 Temperature May Not Simulate Accurately

Putting a non-zero value for the temperature parameter should impose a random field vector on the sample, scaled by the temperature value entered. The scaling may not be implemented accurately, and non-zero temperatures have not been tested. It is recommended, when reasonable, to set the temperature to zero.

## 6.2 Vector Length May Deviate from Unit Value

**Note that by default, vector re-normalization is enabled, to prevent vector length from deviating significantly from unit length. This may in itself produce oddities in simulation as the vector components are re-normalized together.**

For some simulations, particularly if integration precision is reduced, the vector length for magnetization may shrink from 1. This is not allowed by definition as the magnetization is represented by a unit vector, i.e. the length must always be equal to 1. This behavior is a result of cumulative error in LLG integration, due to the limits of machine precision.

To work around this, either increase integration precision by decreasing the values of the *ABS_PREC*, *REL_PREC*, and *PREC_MODIFIER* defines, or un-comment the define *INT_RENORM_VECTOR* (which is uncommented by default). The latter will re-normalize the vector when the vector length drops below *INT_RENORM_VECTOR_TOL.* **Vector re-normalization is enabled by default.**

## 6.3 Simulation May Run For Extended Time

Some simulation runs will fail to converge. In these cases, the integrator may slow each iteration down to the minimum step size. As a consequence computation time for an iteration may be greatly increased.

To work around this, either decrease the precision by increasing the precision values as explained in section 6.2 or increase the minimum step size by increasing the define *INT_MIN_STEP_SZ*. Changing the integrator type by changing the value of the define *INTEGRATOR_TO_USE*; the GSL documentation has more information about what types of integrators are available[18].

Note that if a particular run takes longer than *INT_TIME_SECONDS* a debug message will print stating that a certain simulation run is taking a long time. To inhibit this output comment out the define *INT_TIME_DEBUG*.

[18] See http://www.gnu.org/software/gsl/manual/html_node/Stepping-Functions.html

# 7 Overview of the Code

An in-depth description of the simulator and GUI code is outside of the scope of this manual. Most of the code is commented reasonably well; in the C code, all function headers are described, and in the python, doc strings are used where appropriate. To supplement this documentation, code structure diagrams are given on the next two pages. The next page describes the structure and function of the Python GUI code, with function calls and accesses being drawn as arrows. The page after that describes the structure and function of the C engine code. Function calls and data passing are drawn as arrows. These diagrams may be useful for a programmer going through the Spinsim GSL code in "getting their bearings". However, the diagrams themselves are subjectively organized and not complete; they should only be used as a rough reference. The programmer should always examine the code itself before making changes or additions, or using the Spinsim GSL code in another project, as it is the code itself that defines the behavior of the program.

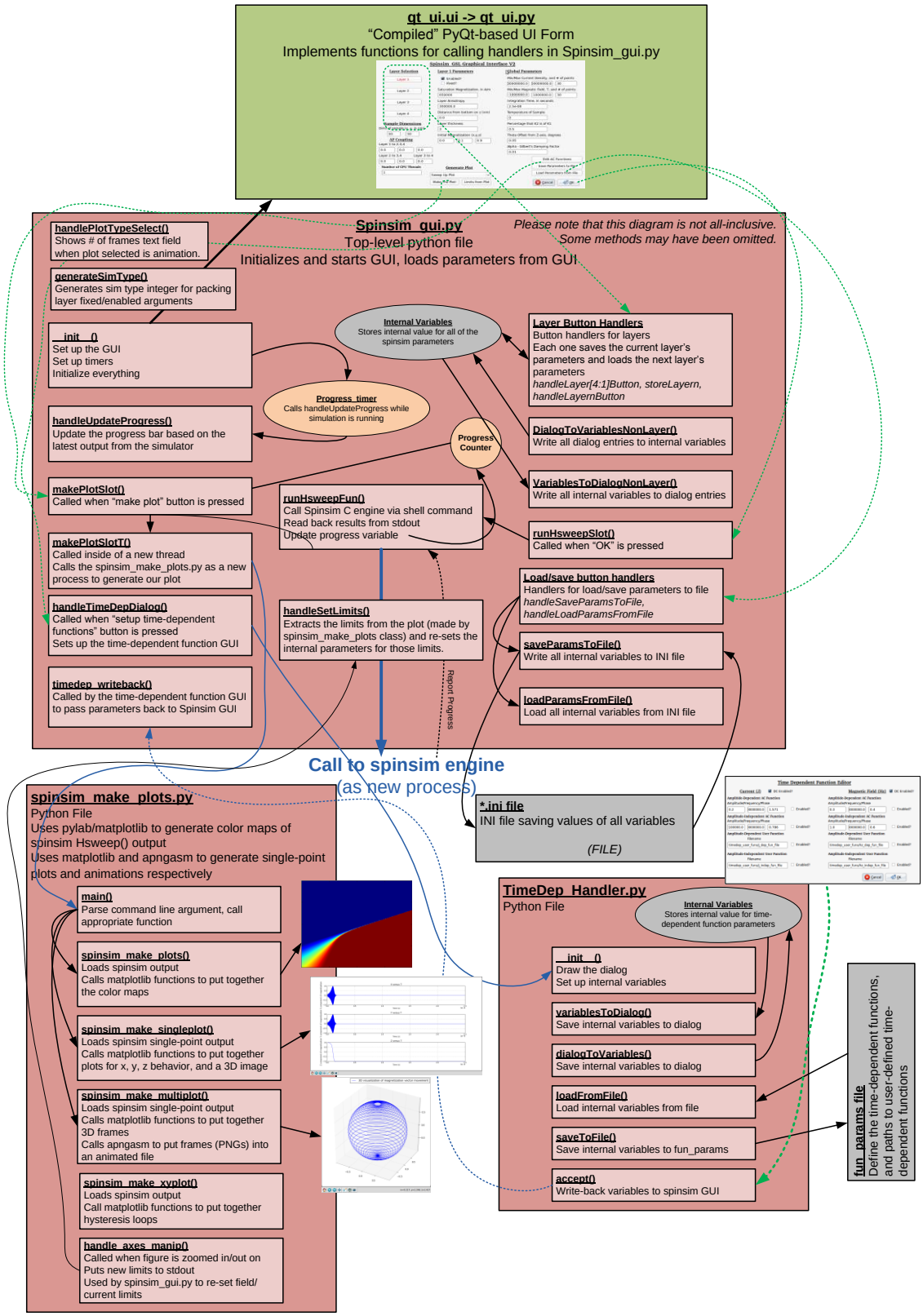



*Figure 19 – Diagram of Python GUI Code Structure*

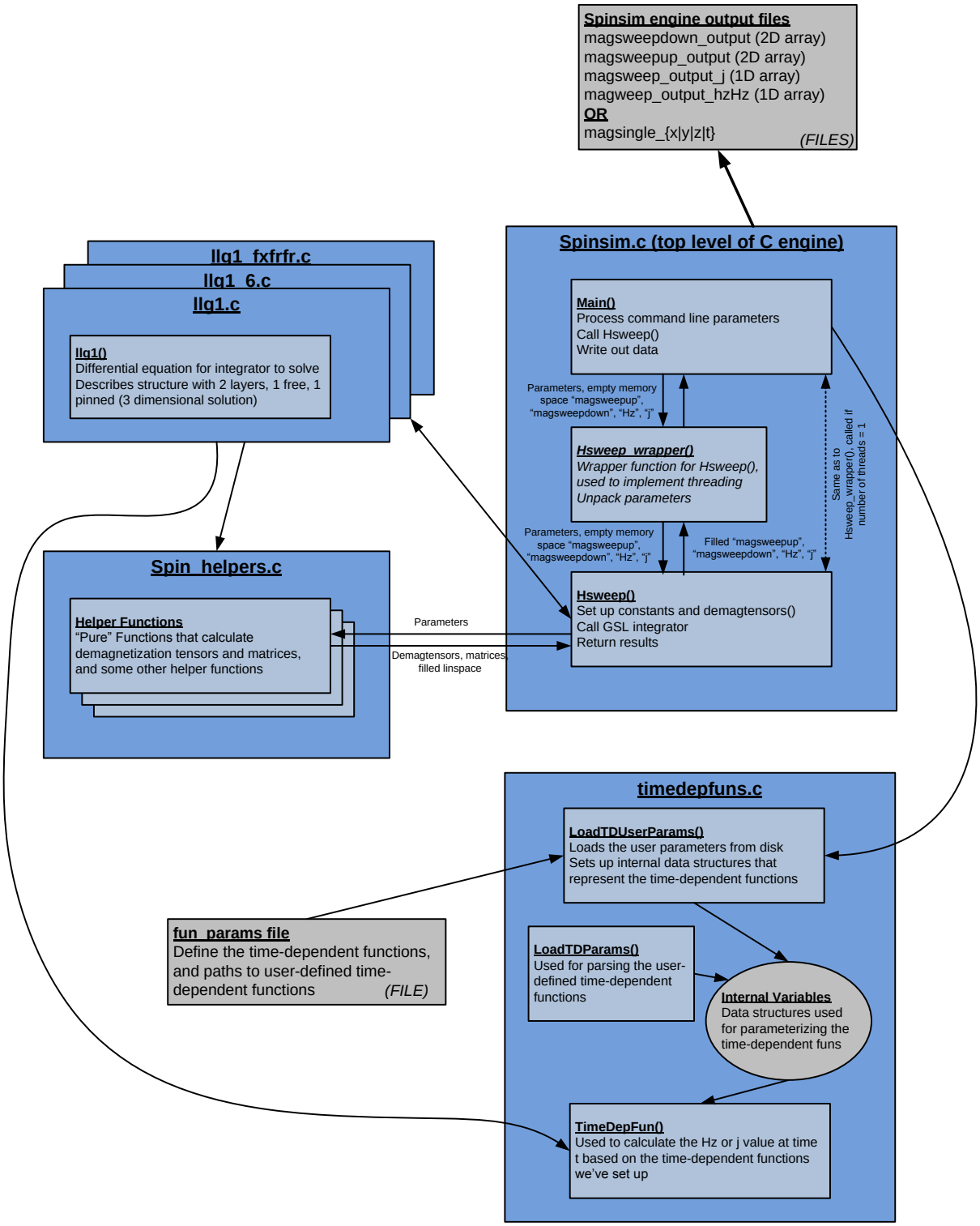



*Figure 20 - Diagram of C Code Structure*