MERRILL

Micromagnetic Earth Related Robust Interpreted Language Laboratory

v1.3.5

May 2019

Contents

1 Important Considerations

1.1 Energy considerations

Energy minimization is limited by physical, numerical and algorithmic constraints.

Numerical constraints The different minimization algorithms have a limited precision. Minimization below this precision can lead to unwanted repetitions due to numerical noise. This can be related to the conjugate gradient method for the sparse matrices or to limited precision in internal functions, e.g. trigonometric functions used for transforming polar to cartesian coordinates.

Algorithmic constraints The finite mesh size or finite distances introduce grid errors and artifacts. The weak FEM solutions are elements of a finite dimensional space approximating a real physical solution. Minimizing below the corresponding approximation error is physically meaningless, even if mathematically correct.

Physical constraints The micromagnetic model only represents a part of the physically relevant energies. Some energy terms are neglected, like magnetostriction, elastic, and electric energies. Most notably thermal activation is disregarded. Minimization of the energy to higher precision than the real variability of the energy is physically irrelevant.

Sometimes it may be mathematically useful to perform higher precision minimizations. This could be necessary to find a complex minimization route that finally leads to a also physically better minimum.

To benchmark different algorithms it also can be useful to compare their output to unphysical degrees of precision.

In most cases it is useful to have a constantly updated estimate of the different accuracy requests to avoid unnecessary minimization steps. The thermal energy per degree of freedom is 1/2kT . In the FEM model the number of degrees of freedom (DOF) is 2 NNODE where NNODE is the number of nodes in the mesh, because each unit vector has two DOF. The total variability in energy density ΔE is then

$$\Delta E = \frac{kT\ NNODE}{V_{tot}}$$

1.2 Meshes

The mesh related discretization error can be estimated by comparing results on two separate meshes with similar number of nodes. To test whether a minimum is really achieved one can monitor largest angular variation along the minimization route. Close to a supposed minimum one can perturb the current state m_0 into a state m_1 which is a distance d away from m_0. If after k minimization steps the result m2 is within ||m2 - m0|| < d/k then m_0 is assumed to represent an LEM.

1.3 Fluctuation field

Van de Wiele et al. (2010) state an expression for the fluctuation field in a micromagnetic model where the amplitude of the thermal fluctuations is derived from the fluctuation-dissipation theorem. A corrected and adapted version of this expression for a tetrahedral subvolume is:

$$H_{therm} = \sqrt{\frac{\alpha k T}{\gamma E \mu_0^2 M_S V_{tet} \delta t}}$$

2 Script Commands

2.1 Introduction

The script is a simple ASCII file containing a sequence of lines. Empty lines, leading, trailing, and multiple spaces or tabs are ignored, as well as anything behind an exclamation mark (!). A number of keywords are used to call subroutines or perform simple assignments. All keywords are case insensitive, e.g. ReadMesh and readmesh are equivalent. The script file is parsed line by line. Each valid line is immediately evaluated and executed.

New or modified functions are highlighted in red.

2.2 Commands

Set <**variable**> <**value**> is used to define global variables for the material, the geometry of the mesh, or program parameters. The following variables are supported:

Ms saturation magnetization in A/m.

K1 1st anisotropy constant for uniaxial or cubic anisotropy in J/m^3.

K2 2nd anisotropy constant for uniaxial or cubic anisotropy in J/m^3.

Aex exchange constant in J/m.

Ls inverse length scale 1/m. Internally Ls^2 is used.

mu related to permeability of free space via mu $= \mu_0/4\pi$.

NEBSpring Spring constant for nudged-elastic band method (NEB) .

CurvatureWeight Weight of curvature contribution for nudged-elastic band method (NEB).

MaxMeshNumber Maximal number of finite element meshes stored. Must be set once before loading meshes.

PathN Number of structures along the magnetization path. *Warning*: The mesh must have been defined previously – Use only after ReadMesh.

ExchangeCalculator Chooses the exchange energy discretization method used. The available choices are $1 = m\Delta m$, $2 = \phi^2$ along edges, $3 = (\nabla\theta)^2 + \sin^2(\theta)(\nabla\phi)^2$.

MaxRestarts Maximum number of restarts during energy minimization.

MaxEnergyEvaluations Maximum number of energy calculations during energy minimization (typical: 10000). Afterwards energy minimization is aborted.

MaxPathEvaluations Maximum number of path energy calculations during path minimization (typical: 2000).

Zone Current Zone to be written into the TecPlot output file (double). Zone can be

set before each output, or can be used with automatic increment.

ZoneIncrement Automatic increment of zone (default=1.0).

SetSubDomain <subdomain number> <variable name> <value>, for multi-phase models. For example, to set Ms to 4.8e5 for subdomain 2 use

SetSubDomain 2 Ms 4.8e5

Note that the SubDomain number must be identical to a block number defined in your PATRAN mesh.

Magnetite <temperature> C Defines material constants for magnetite at temperature <temperature> to be used in subsequent calculations. The temperature is given in degrees Celsius and must be positive and below magnetites Curie temperature at 580°C.

Iron <temperature> C Defines material constants for iron at temperature <temperature> to be used in subsequent calculations. The temperature is given in degrees Celsius and must be positive and below irons Curie temperature at 770°C.

Resize <old length> <new length> changes the length scale of the mesh such that after this rescaling the length <old length> will become <new length> . (cubic — uniaxial) anisotropy Defines the symmetry of the anisotropy energy.

CubicRotation <phi> <theta> <alpha> Performs a 3D rotation of the cubic anisotropy axis. Variables are real numbers that specify the rotation angles in radians. Alpha, theta, phi, corresponds to rotations around the x-, y-, and z-axes, respectively. Positive angles correspond to rotation in a clockwise sense when viewed from the origin along the specified axis.

To rotate [111] to [100]:
CubicRotation 0 0.6154 -0.7854
To rotate [111] to [110]:
CubicRotation -0.4636 0.4205 0.1007

CubicRotation <phi> <theta> <alpha> SD = <Material number where <Material Number> is an integer that applies to rotation to a specified material of a multi-phase model. Alpha, theta, phi, corresponds to rotations around the x-, y-, and z-axes, respectively. Note you need to include the SD = , where SD stands for Sub Domain and a space is needed between SD, =, and the material number. The variable it is set to MUST match a BLOCK ID set in your PATRAN mesh file.

Easy axis <x> <y> <z> Sets the easy axis for uniaxial anisotropy.

Easy axis <x> <y> <z> SD = <Material number> Sets the easy axis for uniaxial anisotropy per subdomain in a multi-phase model.

External field direction <x> <y> <z> Sets the direction vector of an external homogeneous magnetic field. Intrinsically sets the field to B = 1 T.

External field strength <B> <unit> Sets the strength of the external homogeneous magnetic field as B in units of unit. Possible values for unit are muT, mT, or T. Must be set after defining

the direction. Subsequent calls reset the field to B without changing the direction. Can be used for hysteresis modeling.

ReadMesh <index> <filename> [**filetype = <filetype>**] Reads the Patran file <filename>, and stores the corresponding mesh and finite element arrays at location <index>. The index must be less or equal to the previously set MaxMeshNumber.

Optionally filetype = <filetype> can be specified where <filetype> takes the value patran or tecplot. If not specified patran is assumed. Both block and point format tecplot files are supported e.g.,
ReadMesh 1 tecplotfile.tec FileType = tecplot
Note the required space between FileType and =.

LoadMesh <index> Loads a previously read mesh and its finite element arrays from location <index>. This mesh will then be used in subsequent operations.

ReadMagnetization <filename> Reads a magnetization file (.dat or .restart) into the current mesh magnetization array. Make sure that it was created for the currently active mesh! The magnetization read is used in subsequent operations.

Uniform magnetization <x> <y> <z> Creates a uniform magnetization for the current mesh pointing in the normalized direction <x>,< y>, <z>. The optional parameter b is the index of the block of nodes in the mesh that should be set. By definition block 1 contains the free nodes, while higher block numbers can be used to define fixed nodes. These block have to be defined in the Patran file. Any previous magnetization is lost!

Uniform magnetization <x> <y> <z> SD = <Material number> Creates a uniform magnetization for the current mesh pointing in the normalized direction <x>,< y>, <z> where <Material Number> is an integer that applies the uniform direction to a specified material of a multi-phase model. Note you need to include the SD = , where SD stands for Sub Domain and a space is needed between SD and =. The variable it is set to MUST match a BLOCK ID set in your PATRAN mesh file.

Randomize magnetization <angle> Randomly changes each current magnetization vector by at most <angle> degrees. The previous magnetization is (partly) lost!

Randomize magnetization <angle> SD = <Material number> Randomly changes each current magnetization vector by at most <angle> degrees, where <Material Number> is an integer that applies randomisation to a specified material of a multi-phase model. Note you need to include the SD = , where SD stands for Sub Domain and a space is needed between SD and =. The variable it is set to MUST match a BLOCK ID set in your PATRAN mesh file.

Randomize all moments Replaces the current magnetization by randomly distributed unit vectors. Any previous magnetization is lost!

Randomize all moments <angle> SD = <Material Number> Replaces the current magnetization of specified subdomain by randomly distributed unit vectors, where ¡Material Number ¿ is an integer that applies to randomisation to a specified material of a multi-phase model. Note you need to include the SD = , where SD stands for Sub Domain and a space is needed between SD and =. The variable it is set to MUST match a BLOCK ID set in your PATRAN mesh file.

ReMesh <index> Takes the current magnetization array and interpolates it at the nodes of the previously read mesh at location <index>. This mesh from location <index> is then loaded and will be used in subsequent operations.

ConjugateGradient Uses conjugate gradient steps during the accelerated descent.

SteepestDescent Uses normal gradient steps during the accelerated descent.

Minimize Calls the minimization routine for the current mesh and initial magnetization. This call does not save the final result!

EnergyLog <filename> Starts logging all subsequent energy calculations into the logfile <filename>.log. For multi-phase material the magnetization reported at each node is the box volume average of relative Ms*m for each of the elements to which the node belongs. Logging can be stopped by EndLog or CloseLogfile

CloseLogfile Ends the previous logging of energy calculations or path minimizations.

WriteMagnetization <filename> [**<packing>**] Saves the magnetization and the mesh. This produces two files: **<filename>**.dat contains vertex coordinates and magnetization vectors only, and **<filename>**_mult.dat which is TecPlot format file containing the data and the mesh for visualization using ParaView or TecPlot. Contains mesh geometry and one or more magnetization states.

The optional <packing> parameter affects only <filename>_mult.dat output. <packing> takes the value of either BLOCK or POINT. BLOCK format is required to save the sub-domain regions of a multi-phase solution to a Tecplot file (readable by Paraview). The subdomains are numbered by an SD variable for each element. BLOCK is now the default output format. Note that the <filename>.dat is unchanged (point) format and contains no subdomain information.

WriteDemag <filename> [**<packing>**] Saves the internal grain demagnetization field and the mesh: <filename> demag.tec TecPlot file for visualization using ParaView or TecPlot. Contains mesh geometry and demagnetization field for one or more magnetization states. The output is of the form x, y, z, Hd_x, Hd_y, Hd_z, SD Saves in POINT or BLOCK format. Note SD parameter is only output in BLOCK format The demag field is in A/m.
The optional <packing> parameter takes the value of either BLOCK or POINT. BLOCK format is required to save the sub-domain regions of a multi-phase solution to a Tecplot file (readable by Paraview). The subdomains are numbered by an SD variable for each element. BLOCK is now the default output format.

WriteHyst <filename> Saves hysteresis data in 5 columns of MH_{ext}, —H— and the 3 components on the average unit M vector, where M is the magnetization and H_{ext} is the external field. MH_{ext} is normalized to the saturated magnetization in the direction of the applied field. For multi-phase material the magnetization at each node is the box volume average of relative Ms*m for each of the elements to which the node belongs. Output file is <filename >.hyst

WriteLoopData <filename> <Name1> <Value1> <Name2> <Value2>... Save field, moment, total volume and user specified variables to file <filename>.loop. <Name1> specified the

column header printed to the file and must not contain spaces. <Value1> is the real number parameter value to be saved. Useful when looping over user defined variables.
Example usage:
Loop Br -100 10 100
WriteLoopData MyOuputFile Reversal_Field Br
EndLoop

WriteBoxData <filename > Writes the magnetization per node along with the node associated volume x,y,z, m_x, m_y, m_z, vbox. Where x,y,x is the node location, m_x, m_y, m_z is the magnetization unit vector components, and vbox is the volume associated with a node (in units of microns3). This is useful if you want to compute the absolute magnetization associated with each node.

MagnetizationToPath <index > Saves the current magnetization in the path at location <index>. This allows to assemble a path from individual magnetization states that have to fit to the current mesh! After assembling a path it must be renewed before further operations can be performed.

PathToMagnetization <index> Moves the path magnetization state at location <index> to the current magnetization. This allows to change individual magnetizations in the path. E.g. Initial and final states of a path read from a file can be minimized for new material constants.

RenewPath Defines all path variables, like distances and tangent vectors, assuming that all magnetizations have been correctly filled.

RefinePathTo <newlength> Refines the current path to a new number of states by linear interpolation in the magnetization angles. This also resets PathN to the new value and renews the path. Of course, the new number of states can also be less than the previous PathN .

WriteTecPlotPath <filename> [**<packing>**] Exports the current path to a TecPlot file with name <filename> . All states along the path are individual zones in the TecPlotFile.

The optional <packing> parameter takes the value of either BLOCK or POINT. BLOCK format is required to save the sub-domain regions of a multi-phase solution to a Tecplot file (readable by Paraview). The subdomains are numbered by an SD variable for each element. BLOCK is now the default output format.

ReadTecPlotPath <filename> Reads a new path from a TecPlot file with name <filename>. All states along the path are individual zones in the TecPlotFile. Because this also reads in the mesh, all mesh related quantities are recalculated. Make sure that all material parameters are correctly assigned, since those are not read !

ReadTecPlotZone <filename> <zone number> Reads one zone from a TecPlot file with name <filename> and makes this the current magnetization. Can be used to read in an initial guess or as a start/end point of an initial path for NEB calculations. No mesh information is read so make sure the mesh appropriate to the data is already loaded.

AppendTecplotZone <filename> [**<packing>**] Saves the magnetization and the mesh to a file **<filename>**_mult.dat. For the first call to this command, or when the filename is changed, it will write the magnetization and the finite-element mesh. Subsequent calls will write the magnetization only. This is useful for multiple solutions that use the same mesh (such as field or volume hysteresis).

Each Zone can be named using the ZoneName command. Files are written in the TecPlot file format containing the data and the mesh for visualization using TecPlot only. Paraview can currently only reads Tecplot format files that contain only **one** zone, and so files created using this command generally cannot be read by paraview.

The optional <packing> parameter takes the value of either BLOCK or POINT. BLOCK format is required to save the sub-domain regions of a multi-phase solution to a Tecplot file. The subdomains are numbered by an SD variable for each element. BLOCK is now the default output format.

ZoneName <string> Provides a Zone name that is written to any Tecplot format file, so works with command WriteMagnetization, WriteDemag and AppendTecplotZone. The string is limited to a maximum of 80 characters.

KeyPause Pauses script evaluation and waits for Key+Enter for continuation.

MakeInitialPath Assumes that a path is defined by set PathN <number > and that the first and last magnetization patterns are defined. Then proceeds by stepwise minimization to construct an initial path for subsequent optimization by the NEB method.

PathMinimize Assumes that an initial path is defined and minimizes the action integral using a variant of the NEB method.

PathLogfile <filename> Starts logging all subsequent path minimization calculations into three logfiles <filename>.enlog <filename>.grlog, and <filename>.dlog. They contain energies along the path, norms of the gradients along the path and cumulative distances along the path. Logging can be stopped by EndLog or CloseLogfile.

SystemCommand <command>... Performs the system command in the remaining arguments as a line. No guarantee can be given for correct behavior. Uses FORTRANs SYSTEM command.

PathStructureEnergies <filename> Computes the energies for each structure along the minimum energy path. The output is written to <filename>, however if this is omitted then the output is written to standard out.

ReportEnergy Makes a report on the model parameters (mesh size, material parameters, exchange length etc, as well as the component and total magnetic energy both in reduced units of (Kd. V) and in Joules.

(Stop — End) Stops script evaluation.

The following commands are advanced options for scripts using loops or variables.

Loop <variable > <startvalue > <endvalue > [<step>] Takes all commands until the next EndLoop statement and performs a loop over the enclosed commands by replacing the variable <variable> with values from <startvalue > to <endvalue > in steps of <step >. If step is not given stepsize step=1.0 is assumed. Within the loop the string #<variable> is replaced by the integer value of variable, the string %<variable> is replaced by the double precision value of variable , and the string $<variable>$ is replaced by a string of the value of variable . Nested loops are not supported! Warning: The Loop command itself must NOT contain variables! This is so because

currently the parsing for replacing variables is performed only after unravelling the loops.

EndLoop Delimits the set of commands inside the active loop.

Define <variable> <value > Defines a numeric variable that can be used like a loop variable.

AddTo <variable> <value > Adds a number to a previously defined variable.

Undefine <variable> Forgets the previously defined variable.

3 Compiling MERRILL

This section is just for developers. If you just want to use MERRILL, precompiled binaries are available for LINUX, macOS and Windows, and can be downloaded from the MERRILL homepage at: `http://www.rockmag.org` .

If you want to modify the way MERRILL behaves, then you will need to download the source code from the repository at:

`https://bitbucket.org/wynwilliams/merrill-1.35`

Every computer and operating system is unique and software and compilers may operate or install differently on different systems. These guidelines are there provided for general guidance, but should work in the majority of instances.

3.1 Windows

3.1.1 Requirements

This approach uses the MinGW (GNU ports) of **gcc/gfortran, gmake** and **cmake** for Windows. Most users will need to install these packages as they are not usually installed on Windows.

1. Install MinGW (GNU port of gcc/gfortran for Windows). The basic default basic install is fine.

   `http://www.mingw.org`

2. Set the environment variables to add the compilers to the Windows Command Line or PowerShell (I like the new PowerShell in recent versions of Windows, but you can use the standard cmd.exe if you prefer.)

   (a) This step varies slightly depending on your version of Windows (7/8/10), but you are looking for the **Environment Variables**, which is usually found in the **Advanced Settings** of the System Properties dialogue box, in the control panel or by right-clicking on the 'Computer' desktop or start menu item. A useful visual guide is here:

   `https://www.rose-hulman.edu/class/csse/resources/MinGW/installation.htm`

   (b) In the **Environment Variables** dialogue box, look for the **System Variables** box, select the **Path** variable, and click **Edit**. (Do not delete anything!)

   (c) Navigate to the end of the Edit System Variable dialogue box, and append the following to the very end of the Variable Value for Path: (Assuming the location of the MinGW installation is here)

   C:\MinGW\bin

   (d) Click OK and Apply the settings.

   (e) In a shell or command line, the gcc command, and related commands should now be available.

3. Install CMAKE
   (a) `https://cmake.org/download`
4. Create a build folder in the Merrill directory and change into that directory (we called it "build")
5. Run CMake from within that build directory just created
   (a) Note we are not using the unix.sh script shipped with Merrill
   (b) Remember to specify the above directory with "..", as we are using the CMakeLists.txt file in the top-level MERRILL directory, one directory above us now.
   (c) The -G flag specifies the generator system name. We want to use the MinGW system that we have previously installed.
   (d) A full list of available generator systems is displayed by running cmake –help

3.1.2 Compile commands

1. (a) To make a **dynamic build** (the usual option, creates small build size of about 0.5Mb but needs runtime libraries to loaded dynamically) use:
   `cmake -G "MinGW Makefiles" ..`
   the two dots at the end are important!
   OR
   (b) To make a **static build** (bigger build size of about 1Mb, but easily moved to other windows machines):
   `cmake -G "MinGW Makefiles"  -DCMAKE_EXE_LINKER_FLAGS:STRING="-static"`
   `-DCMAKE_BUILD_TYPE:STRING="Release" -DBUILD_SHARED_LIBS:BOOL=OFF ..`
   remember the two dots at the end!

   These create a Makefile in the build directory.
2. We now issue the make command to use this generated makefile:
   `mingw32-make`
   Which will produce the **merrill.exe** executable file and libmerrill.a. In the case of a static build the libmerrill.a is not needed for merrill.exe to run.

3.2 OS X

3.2.1 Requirements

1. Install Command Line Tools
   This requires the MacOS App Xcode installed. Xcode contains the tools needed to compile native code on your machine. It can be downloaded from the App Store or from `https://developer.apple.com/xcode/downloads/`.

This is a large app and can take sometime to download and install. Once installed open an terminal window and and execute the command

```
xcode-select install
```

and press enter/return. You may be prompted for your password. Then wait for the command to finish.

2. Install HomeBrew.

   Homebrew makes it very easy to install and maintain various command line tools and is highly recommended if you intend to any software development. To install open a terminal window and execute:

   ```
   ruby -e "$(curl -fsSL
   https://raw.githubusercontent.com/Homebrew/install/master/install)"
   < /dev/null 2> /dev/null
   ```

   type the full text and then press enter/return. You may be prompted for your password.

   If you get an error "Ivalid certificate chain" when downloading from github, the following will usually solve this. At the terminal prompt type:

   ```
   curl https://raw.githubusercontent.com/Homebrew/install/master/install
   ```

   and press enter/return. Then re-execute the previous command:

   ```
   ruby -e "$(curl -fsSL
   https://raw.githubusercontent.com/Homebrew/install/master/install)"
   < /dev/null 2> /dev/null
   ```

3. Install cmake.

   At the terminal prompt type:

   ```
   brew install cmake
   ```

   and press enter/return.

4. Install gfortran (if necessary)

   Xcode will normally install gfortran. You can test this by typing `gfortran` at the terminal window command prompt. If installed it will complain `gfortran: fatal error: no input files`, and your good to go. If it is NOT installed it will complain `gfortran: command not found`, and you need to install gfortran, by issuing the following command at the terminal window prompt:

   ```
   brew install gcc
   ```

   and press enter/return.

3.2.2 Compile commands

1. In directory Merrill create a directory **build**, will contain the compiled code.
2. (a) For a **dynamic** build (the usual option, creates small build size of about 0.5Mb but needs runtime libraries to loaded dynamically):

   From the build directory issue the command:

   ```
   cmake ..
   ```

   note the two dots at the end are important.

   When that comand has finished , issue the build command:

   ```
   make
   ```

   This should compile and link and create an executable file called **merrill**.

   OR

   (b) To make a **static build** (bigger build size of about 1Mb, but easily moved to other macOS machines), issue the commands:

   ```
   export FFLAGS= "lz static-libgfortran static libgcc"
   ```

   then

   ```
   cmake ..
   ```

   and finally

   ```
   make
   ```

3. Running merrill.

   It is useful to have the location of the executable code in your path. This can be done by ensuring that the path of the directory is listed in your environment variable **$PATH** stored in your **.bash_profile** or **.profile** file in your home directory. It should contain a line such as

   ```
   export PATH="/Users/wynwilliams/merrill/build:$PATH"
   ```

   you can n then issue the command

   ```
   merrill <scriptfilename>
   ```

   from any directory

3.3 Linux

Congatulations! your already a computer expert and can figure out what do. (similar to the macOS example above)

3.3.1 Requirements

3.3.2 Compile commands