MAGE

J. Chęciński, M. Frankowski

April 16, 2016

Program structure and principles

MAGE is a graphic user interface program meant to assist end users with the creation of both OOMMF configuration files (.mif files) for simulations and MATLAB files for the analysis of the obtained results. The program consists of two modules, the first one responsible for the OOMMF part and the second one responsible for the MATLAB part.

In order to run any of the modules, you can use the MATLAB environment and run the respective .m file (mage.m or mfile_generator.m) from the MATLAB window. Alternatively, you can use the GUIDE addition to MATLAB and run these files via GUIDE. The first part of MAGE is partially independent from the MATLAB environment and can be run as a standalone program (mage.exe) provided that you install the respective MCR package provided by Mathworks company for free. Since the second part requires a working MATLAB environment in order to make use of the analysis files, it cannot be run separately as a standalone program.

Mif generation module

The first module consists of a main window and several buttons that will redirect you to subwindows allowing for creation of .mif code. For a description of the OOMMF classes, see: http://math.nist.gov/oommf/doc/userguide12a5/userguide/Standard_Oxs_Ext_Child_Clas.html. Generally, the list of MAGE features corresponds to the list of OOMMF classes and the user should refer to the description linked above in order to understand and make use of the program. Some of the less general and rather uncommon classes are not fully supported by the current version of MAGE. If you would like to add one of them to your file, you have to introduce the changes manually into the final .mif file or into the program edit window. Additionally, the program supports a number of classes not included in the basic OOMMF list such as an evolver allowing for an introduction of spin-polarized current or a magnetoresistance class. For the detailed description of these, see paragraphs subsections 3.1.5 and 3.1.6.

The GUI of MAGE Mif module main window is presented in Fig. 1.

New Mif

'New Mif’ button creates new mif header, asking the user for confirmation first if there is already an existing one. This button has no additional options or parameters.

Atlases

'Atlases’ button is responsible for creation of Oxs_Atlas classes. There are five possible types of atlases: rectangular cuboid, ellipsoid and three types of disc-like cylinders (with x, y or z axis being the main axis). The rectangular cuboid case is realized as an Oxs_BoxAtlas object, the other cases are realized as Oxs_ScriptAtlas objects with scripts being generated automatically by the MAGE program. The user
has only to specify the desired type of the atlas, its size (x, y and z span in nanometers - must be a
positive value), the name of the multiatlas used and optionally also the name of the atlas being created.
Note that MAGE supports only the kind of simulation architecture where created atlases are put inside
a multiatlas, effectively becoming the regions of this multiatlas. If you are not sure how to use different
multiatlases, it is recommended to leave this option unchanged.

One more Oxs_Atlas class which is also supported by the program is the Oxs_ImageAtlas, which
can be created in a separate subwindow. The parameters you need to specify here are the same as in the
standard Oxs_ImageAtlas class description.

Once you define the desired parameters for the creation of your atlas, click ‘Add This Atlas’ button
in order to add it to the list of atlases visible in the right part of the subwindow. If you made a mistake,
you can remove any entry from this list using the ‘Remove Selected Atlases’ button. Only the atlases
included on the list when you click the ‘Confirm and Quit’ button will be actually added to the mif
content. Alternatively, you can use the ‘Cancel and Quit’ button in order to exit without introducing
any changes to the mif file being created. This type of architecture (a listbox containing multiple and
potentially removable entries representing elements of the same type being created, cancel and confirm
buttons) will be also used in the next subwindows.

Overview of ‘Atlases’ window is depicted in Fig. 2.

**Scalar and Vector fields**

‘Scalar and Vector fields’ button is responsible for creation of Vector and Scalar Field Objects. The field
objects which can be created in MAGE are listed under the ‘Field Type’ button group:

- Oxs_UniformScalarField,
Figure 2: GUI of MAGE 'Atlases' window.

- Oxs_AtlasScalarField,
- Oxs_RandomScalarField,
- Oxs_LinearScalarField,
- Oxs_UniformVectorField,
- Oxs_AtlasVectorField,
- Oxs_RandomVectorField,
- Oxs_PlaneRandomVectorField.

Each of them has its own unique set of parameters to be defined, in accordance with their .mif pattern. In the case of Atlas Fields, you have to make use of the already existing atlases in order to create a list of regions with their respective values. If you are defining a vector field, you can also choose to include an additional Oxs_FixedZeeman object together with it with a given multiplier value.

**Internal energies**

'Internal energies' button is responsible for creation of Energy objects, namely time-independent anisotropies: Oxs_UniaxialAnisotropy and Oxs_CubicAnisotropy), exchange interactions (Oxs_Exchange6Nzgb, Oxs_Uniform Exchange, Oxs_ExchangePtwise, Oxs_TwoSurfaceExchange and Oxs_RandomSiteExchange) as well as Oxs_Demag. Oxs_Zeeman classes are not included here, since they will be dealt with separately in the Excitations subwindow.

The Anisotropy subwindow allows you to create a Oxs_UniaxialAnisotropy or Oxs_CubicAnisotropy anisotropy block. An existing atlas block is necessary. Both the anisotropy direction and its value can be chosen either as a constant custom value or as a vector/scalar field, if it has been already created. Additionally, you can decide whether the anisotropy constant or the anisotropy field will be used in the block definition. Note that, in the case of cubic anisotropy, the specified axes must be orthogonal to each
other. MAGE performs such an orthogonality check for custom directions, but not for existing vector fields.

The interactions subwindow allows you to create one of the five interaction types. Each of the types contains a number of necessary or optional parameters in accordance with their OOMMF definitions. Note that you need to have an existing atlas block in order to create an Oxs_Exchange6Ngbr class object, an existing scalar field in order to create an Oxs_ExchangePtwise object and both atlas and scalar field blocks in order to create an Oxs_TwoSurfaceExchange class object. Oxs_UniformExchange and Oxs_RandomSiteExchange objects do not have any similar requirements.

Since Oxs_Demag class in OOMMF does not use any parameters, the user can only decide to include it (or not), using the Include Oxs_Demag checkbox in the main Internal Energies window.

Excitations

'Excitations' button is responsible for the creation of excitations: magnetic field (through Oxs_ScriptUZeeman objects), voltage-controlled magnetic anisotropy (through Oxs_UniaxialAnisotropy objects) and electric current (through MF_CurrentFlowEvolver objects). You may choose between those three types of excitation and between five basic excitation shapes, which can be subjects to further sweeping if desired. We will now briefly review the excitation shapes and the sweeping options.

• 'Sinusoid': a sine-type excitation described by following formula:

\[
h(t) = A \sin(2\pi ft + \phi) + B. \tag{1}\n\]

Coefficients \(A\) and \(B\), as well as the frequency \(f\) and phase \(\phi\), are defined by the user, and \(t\) is the simulation time. Note that frequency is specified in GHz, phase in degrees and the unit of \(A, B\) coefficients can be \(mT\) (magnetic field case), \(kJ/m^3\) (anisotropy change in case of electric field) or \(mA\) (electric current case).

• 'Square': basically, a sign function of the sine-type excitation described previously. Instead of a sine, the final result is a square function here, with all its parameters and coefficients defined in a way similar to the one used in the sinusoid type of excitation. Additionally, the user can specify the slope width parameter which causes the square slopes to behave in a realistic way as they no longer switch from high to low (or from low to high) state immediately. For example, if the slope width is specified, the rising slope of the excitation will be given by a quadratic formula:

\[
h(t) = A \left(1 - \left(\frac{t - T/2 - s}{s}\right)^2\right) + B, \tag{2}\n\]

where \(A\) is the amplitude, \(B\) is the constant part of the excitation, \(s\) is slope parameter (in seconds) and \(T\) is the square period (calculated automatically from the frequency specified by user).

• 'Pulse': a single gaussian-like pulse excitation described by the formula:

\[
h(t) = B + A\exp\left(-\frac{(t - t_0 - \tau/2)^2}{\tau^2/32}\right). \tag{3}\n\]

The user can regulate the amplitude, offset \(t_0\) and the width \(\tau\) of the pulse. The width is calibrated in such a way that the value of the non-constant part of the excitation \(h\) is less than 0.1\% of the maximum value outside of the range \([t_0, t_0 + \tau]\). The offset always refers to the beginning point of the pulse, not to its center.

• 'Step': a single step which is similar to the square-type excitation in shape, but occurs only once just like the pulse-type excitation. The parameters used here are the same as in the pulse-type excitation, with width having a straightforward interpretation of width of the step used. Similarly to square-type excitation, the user can also define a slope width parameter here.
• 'Fixed': a simple constant value is returned. The user can only define the amplitude.

Apart from five basic shapes, there is also a number of possible sweeping options available for the user:
• 'No sweep': default option where no sweeping is performed at all.
• 'Repeat': here, the excitation is repeated every given period of time (defined by the user in the Repeat each field) without any parameter changes. Sinsuoid and square-type excitations are largely unaffected by this option, except for the fact that their phase will reset to the initial value each time a new repeat period is started. It can be however useful for defining multiple pulses or steps of the same strength and width.
• 'Amplitude': an amplitude sweep where every repeat period has its own amplitude. The user defines the starting value, the value by which the amplitude changes each cycle, and the maximum end value which the amplitude cannot exceed. For example, choosing 10 mT as the starting value, 15 mT as the change by value and 50 mT as the end value will result in the amplitude being equal to 10, 25, 40, 50, 50, ... mT in the consecutive excitation cycles. Note that the sweep can be performed from higher values of amplitude to lower ones if you define the change by value to be negative. In such a case, the end value represents the possible minimum and not the maximum.
• 'Frequency': a sweep similar to the one described previously, except this time the frequency is subject to change. This sweep is available only for sinusoid and square-type excitations.
• 'Phase': a sweep similar to the one described previously, except this time the initial phase is subject to change. This sweep is available only for sinusoid and square-type excitations.
• 'Offset': a sweep similar to the one described previously, except this time the initial offset is subject to change. This sweep is available only for pulse and step-type excitations.
• 'Width': a sweep similar to the one described previously, except this time the width is subject to change. This sweep is available only for pulse and step-type excitations.

Apart from excitation types and sweeps, each entry in this window also contains the start time and end time options. An excitation is only active between those time points, and its value outside of the specified range is always equal to zero. If the start time value is other than zero, the excitation will wait until the start time before calculating e.g. initial offset values. For example, if a magnetic pulse is defined with start time equal to 10 ns and offset equal to 8 ns, the actual pulse will occur at 18 nanoseconds from the start of the simulation.

All magnetic field excitations have an additional option, which is the direction of the magnetic field applied. The direction can be static, or swept during the excitation time. If the direction is swept, the initial value is taken from the main Parameters panel and the user has also to define the rotation axis as well as the rotation angle (in degrees) with respect to this axis. For example, setting the initial direction to [1 0 0], the rotation axis to [0 0 1] and the rotation angle to 45 degrees will cause the field unit vector to change direction from [1 0 0] to [1 1 0], then to [0 1 0] etc. Note that, in the case of direction sweep, there is no end value to possibly define here.

In the case of voltage or electric field excitations, the user has also to specify excitation details, namely the uniaxial anisotropy or the MF_CurrentFlowEvolver block. The way of defining the uniaxial anisotropy is the same to what has been described previously. The MF_CurrentFlowEvolver is commented on in more details below, in the Miscellaneous window description.

Magnetic and electric field excitations can be applied multiple times in the same simulated system. However, this is not the case with current excitations, since in OOMMF architecture they are implemented as a part of evolver (here, MF_CurrentFlowEvolver). As a result, although the user can define multiple current excitations using MAGE software, in the simulation only one of them can be used at a time, depending on which evolver has been chosen by the driver object to run.

The GUI of 'Excitations' window is presented in Fig. 3.
Miscellaneous

'Miscellaneous' button is responsible for creation of meshes, evolvers, drivers or custom analysis blocks. The user can choose whether to include any of those. The creation of mesh is relatively simple and requires only a specification of multiatlas and x,y,z step sized. Optionally, mesh name can also be defined.

There are five possible evolvers which can be created: EulerEvolve, RungeKuttaEvolve, CGEvolve (energy minimalization evolver), SpinXferEvolve and MF_CurrentFlowEvolver. The first four are incorporated in the basic OOMMF version and described on the OOMMF website. The last one is an additional extension which has to be compiled with OOMMF code independently from MAGE tool. It can be defined in the previously described 'Excitations' window for spin-transfer-torque effects in magnetic tunnel (or GMR) junctions. In such a case, the program will automatically identify an already defined block and suggest not to specify any further evolvers.

Regarding the common parameters, majority of them could usually be left empty (leading to default OOMMF values). The only exception is the damping parameter alpha, which has to be defined for all time-domain evolvers. Also, if the user wishes to perform Fourier analysis of simulation results later, min and max timestep values (available, similarly to other situational parameters, in the 'Advanced' window) should be set equal to each other.

MF_CurrentFlowEvolver is an extended version of old RungeKuttaEvolve implementation and uses nearly the same parameters as well as the following new current-related parameters: out-of-plane torque
coefficients bJ0, bJ1 and bJ2, resistances of parallel and antiparallel states R_P and R_AP, effective spin polarization eta, junction_surface parameter (only to be used in rare case of unsymmetrical surfaces of junction interfaces), current_mode that defines the type of the electric signal source: 0 for a voltage source (resistance-dependent inhomogeneous current), 1 for a current source (resistance-independent inhomogeneous current) and any other value for voltage source with constant and homogeneous current.

In order to create a MF_CurrentFlowEvolver class, it is necessary to define the interfaces of both layers which will be used for calculation of resistance (due to magnetoresistance effect) as well as spin transfer interaction between polarized current and magnetization. Therefore, the user needs a valid atlas block and a scalar field to introduce surfaces in a manner similar to the case of Oxs_TwoSurfaceExchange class.

Last but not least, the evolver contains a handle for a script that defines time-dependence of used signal (required). However, to take advantage of MAGE support for automatic script generation, one has to use 'Excitations’ window to take care of that parameter (otherwise, the script has to be pre-defined manually).

A driver should be chosen accordingly to selected evolver, e.g. MinDriver for CGEvolve (or other manually added minimization evolver) and TimeDriver for Landau-Lifshitz-Gilbert equation-based approaches. Advanced parameters of drivers can again usually be left empty. Note that if you used the 'Excitations’ window to introduce excitations, the program will automatically suggest a matching stopping time condition (in the case of multiple excitations, the longest one is considered).

In order to set the saturation magnetization and the initial magnetization configuration, the user should use the Magnetization subwindow. Each panel in this window is dedicated to other method of magnetization setting: uniform, atlas-specified or script-specified. The chosen panel is activated by right-clicking on the field in which the parameters are to be provided.

Furthermore, there are two additional extensions, also to be compiled with your OOMMF distribution in order to be used. 'Magnetoresistance’ allows for calculations of giant or tunnel magnetoresistance in junction in exactly the same manner as in MF_CurrentFlowEvolver, but without introducing an evolver and without current and spin-torque calculations. 'Local Magnetization’ subwindow enables simple probing of magnetization averaged over any cuboidal area (for example from one layer of magnetic tunnel junction) in addition to standard OOMMF outputs. Cells with zero magnetization are omitted in localized probing.

Schedule Outputs

'Schedule Outputs’ button is responsible for creation of output parameters schedule output commands. The user can specify a Set Options block with standard MIF 2.2 parameters, especially basename and output format parameters. The default formats are binary (binary 8) for fields and 17-characters precision text for scalar output (for example, .odt file). Text formatting strings are not tested by MAGE at all, so caution is advised. Additionally, the user can quickly include three of the most frequently used schedule output commands (DataTable, Oxs_TimeDriver::Magnetization - if a time driver was specified, and Oxs_Demag::Field - if Oxs_Demag was specified). Custom schedule commands can also be added, but the user has to introduce their names manually.

Menu bar

MAGE features a short menu in the upper left part of the program window. It includes six commands:

- 'create a new mif’ has the same functionality as the 'New MIF’ button (see par. 3)
- ‘load’ and ‘save mif’ commands call for standard MATLAB I/O interface in order to enable loading and saving .mif files
• 'reset warning settings': there is a number of possible warnings in MAGE, e.g. concerning the automatic name control (see next paragraph). The user can discard these warnings by checking 'Do not show this message again’ option. This button allows for resetting of warning settings by restoring all messages.
• 'info' displays a brief information about MAGE program and provides contact to the authors.
• 'exit' terminates the program; it has the same functionality as using the ordinary close button in the upper right part of the program window.

Limitations

MAGE provides a name control for the defined OOMMF classes in order to avoid duplicates which would cause the simulation to return error. The names of the created objects are parsed dynamically, which means that a block of code containing e.g. a scalar field should be recognized by the program even if it was created externally and pasted into the MAGE window. However, in order to support this kind of solutions, there are several limitations concerning the names of the object that can be created. Most of them are already implemented in MAGE during the creation of classes; for example, the program will not allow the user to include a forbidden character into a name. This safety check is not possible when pasting the code from the external source, though, so caution is advised. The full list of limitations is included below and contains:

• forbidden characters ? : " * / \
• keywords: archive, signal_profile, script and multscript. Note that you can use e.g. the word 'script' as a regular keyword in your code even if this code was created externally, but you should not include it as a part of the name of any OOMMF class object.
• if you force the names, especially of multiatlases and scalar/vector fields, to be empty, it may cause the program to not work properly.
• if you add a TCL procedure and a script which will call it, it is highly recommended for the procedure to be placed directly before the script; otherwise, the control of procedure names may not work properly.

Other features

Both commas and dots are recognized by the program as a valid decimal mark. The CTRL button can be used in most listboxes in order to select multiple entries. The program also supports CTRL+S shortcut in order to save your .mif file. The current version of MAGE does not support CTRL+F or CTRL+Z shortcuts. CTRL+C and CTRL+V shortcuts are available by default in MATLAB-type GUIs.

M-file generation module

The second part of the program assists the user in creation of .m files which would allow for analysis of simulation results. It requires a working MATLAB environment in order to make use of the analysis files and cannot be run separately as a standalone program. It consists of only one main window which can later call the created .m scripts. The module allows for easy selection of data columns from .odt files and performing user-defined operations on them. One can either select spectral analysis operations provided in GUI or type in any Matlab commands for evaluation. In the most common case it may be a simple arithmetical operation between selected columns and/or constants introduced by user, similarly to the example presented in the next section. The spectral analysis includes fast Fourier transform (FFT) and wavelet transforms, useful for analysis of dynamics in ferromagnetic nanostructures [?]. Each can be performed separately for a selected time period or periods, allowing for analysis of data obtained by e.g. changing the simulation external condition such as the magnetic field. The user can define several
calculation tasks and the program will generate suitable m-file to execute them. The user can select to save tasks to files, to run the tasks immediately or to do both. Results of the analysis can be plotted, saved to text files or to image files with graphs.

The GUI of MAGE m-file generation module is presented in Fig. 4. The elements of this part of the program will be now briefly described.

![Figure 4: GUI of MAGE m-file generator module window.](image)

Choosing data

'Select .odt File Name or Directory' editable box and 'Browse' button both have the same functionality of choosing the .odt file to be analysed. The user can either write down the path to file or directory manually, or use the browsing window built-in MATLAB to choose them. Note that only the last inserted result will be taken into account by the program.

If the user chooses a single .odt file, the names of the consecutive columns are derived from the file content and displayed in the 'Available Column List' window below the browse button. Each column is also assigned a unique letter tag in order to enable easier navigation in the later parts of the program.

If the user chooses a whole directory, it is scanned recursively (depth-first search) in order to find all .odt files. The first .odt file found is used to load the column list. For the rest of .odt files, in the case that the total number or the names of detected columns differ between .odt files found, only the files with format exactly matching the one of the first .odt file will be accepted and the rest will be ignored. The final, refined list of .odt files should have identical format and allow for applying the same set of defined operations on all files consecutively.

Note that both multiple .odt files and multiple directories can be chosen at the same time; the only limitation is the fact that one cannot choose simultaneously a file and a directory.
Configurating Tasks

The ‘Analysis Task Parameters’ panel allows the user to specify the desired parameters of the analysis. The basic formula of the program is to define one series of .odt data as a function domain x and another one as its set of values y(x). The obtained result can be then subjected to FFT/wavelet operation or simply plotted.

By clicking on a column displayed on the available columns list and then by clicking the ‘Set selected column as y’ (x) button, the user can choose any column as x or y data (for example, magnetic field amplitude can be chosen as y and time can be chosen as x to obtain magnetic field vs. time dependence). Alternatively, arithmetic or custom functions can be used on the selected columns. For example, if you would like to obtain a sum of amplitudes of Zeeman fields denoted on the available columns list as G and K, it is sufficient to insert ‘G+K’ into the editable window and click the respective set evaluation result button. Similarly, operations of subtracting, multiplying, dividing and exponentiation can be performed. The user can also use custom functions or operations - for example, writing ‘my_function(G)’ into the window will set the result of the my_function operation on column with tag G as the desired outcome.

For the same reason, it is not possible to use lowercase instead of uppercase letters as column tags, e.g. g+k instead of G+K will not work properly.

Once the expressions for x and y are set, the user can choose what kind of operation they wish to perform on the gathered data. There are three basic types of operations supported: none (e.g. for simply plotting the function), FFT and wavelet. The additional parameters for the last two operations include:

- offset: the number of initial nanoseconds to ignore before the start of Fourier or wavelet transform
- range: the number of nanoseconds to include in the transform domain
- repeat each: if the transform is meant to be performed cyclically (for example, because one of the parameters of the simulation was swept), this parameter will be responsible for how often the operation is repeated
- frequency limit (FFT only): frequencies above this limit will be discarded in the FFT output; the order of magnitude of the frequencies typically encountered in micromagnetics usually does not exceed several GHz, so this option may be used to keep the output clear from very high frequency noise
- wavelet type (wavelet only): the chosen wavelet (depends on what wavelets are available in the used Matlab instance, see wavemngr Matlab function)
- wavelet scales (wavelet only): 1-D vector of positive values necessary for cwt Matlab function

For example, setting offset to 5 ns, range to 20 ns and repeat each to 30 ns will result in the cycle of transforms performed on the following periods of the simulation: from 5 to 25 ns, from 35 to 55 ns, from 65 to 85 ns ... etc. If the active period of the analysis is empty, the program will treat it as an error and not execute. Note that, for the sake of FFT and wavelet analysis, it is assumed that the x data is a monotonically increasing time vector; it is also used to automatically calculate the sampling frequency. Using other .odt columns as the x data for the FFT-type analysis is allowed, but the program is not guaranteed to run properly. This is especially important if one intends to analyse a simulation file where a MinDriver and not a TimeDriver was used. Finally, it should be also pointed out that the results of FFT created by MAGE software automatically filter the constant coefficient at 0 GHz.

The results of the first two analysis types may be handled in three different ways: they can be plotted in a figure, saved to a .fig file, or saved as a .txt data file. These options are not mutually exclusive, and the user can specify any combination of them using the ‘Results Handling’ subpanel.

Wavelet operation results can only be saved in a .mat file for further external processing. In such a case, the .mat file will consist of a structure containing a time matrix and a structure of coefficient
matrices for each analysis interval (defined by 'range' and 'repeat' parameters').

**Handling results**

As the definition of task parameters is finished, the task can be added to the list visible in the right part of the window. As usually with MAGE lists, its elements can also be deleted at will. There is only one part of the analysis file creation remaining, namely determining the output directory for .m file and data or figure its further results. The output directory is shared between tasks and can be set both manually and via 'Browse' button. The default value is a subfolder 'Results' located in the current_directory. The output directory parameter concerns the created .m file; the figure and data files produced as the result of the analysis are by default located in the same folder where their original .odt file was detected (this is especially useful if one plans to analyse multiple .odt files located in different subfolders at a time). Alternatively, the user can demand that the specified output directory should also be used for figure and text files.

Finally, the user can press the 'Save and Quit' or 'Save' button in order to save the defined tasks to the .m file. The 'Cancel and Quit' button can be used to simply quit the program instance without producing any .m files.