OOMMF
User's Guide

August 10, 1999

This manual documents release 1.1a1.

WARNING: In this alpha release, the documentation may not be up to date.

Abstract

This manual describes OOMMF (Object Oriented Micromagnetic Framework), a public domain micromagnetics program developed at NIST. The program is designed to be portable, flexible, and extensible, with a user-friendly graphical interface. The code is written in C++ and Tcl/Tk. Target systems include a wide range of Unix platforms, Windows NT, and Windows 95/98.

Contents

Disclaimer

1 Overview of OOMMF

The goal of the OOMMF[1] (Object Oriented Micromagnetic Framework) project in the Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) is to develop a portable, extensible public domain micromagnetic program and associated tools. This code will form a completely functional micromagnetics package, but will also have a well documented, flexible programmer's interface so that people developing new code can swap their own code in and out as desired. The main contributors to OOMMF are Mike Donahue and Don Porter.

In order to allow a programmer not familiar with the code as a whole to add modifications and new functionality, we feel that an object oriented approach is critical, and have settled on C++ as a good compromise with respect to availability, functionality, and portability. In order to allow the code to run on a wide variety of systems, we are writing the interface and glue code in Tcl/Tk[2]. This enables our code to operate across a wide range of Unix platforms, Windows NT, and Windows 95/98.

The code may actually be modified at 3 distinct levels. At the top level, individual programs interact via well-defined protocols across network sockets. One may connect these modules together in various ways from the user interface, and new modules speaking the same protocol can be transparently added. The second level of modification is at the Tcl/Tk script level. Some modules allow Tcl/Tk scripts to be imported and executed at run time, and the top level scripts are relatively easy to modify or replace. At the lowest level, the C++ source is provided and can be modified, although at present there is no documentation detailing this process.

The first portion of OOMMF released was a magnetization file display program called **mmDisp**. A working release[3] of the complete OOMMF project is now available. This includes a problem editor, a 2D micromagnetic solver, and several display widgets, including an updated version of **mmDisp**. The solver can be controlled by an interactive interface (Sec. 9), or through a sophisticated batch **control** system (Sec. 16).

The solver is based on a micromagnetic code that Mike Donahue and Bob McMichael had previously developed. It utilizes a heavily damped Landau-Lifshitz ODE solver to relax 3D spins on a 2D mesh of square cells, using FFT's to compute the self-magnetostatic (demag) field. Anisotropy, applied field, and initial magnetization can be varied pointwise, and arbitrarily shaped elements can be modeled. We are currently working on a full 3D version of this code suitable for modeling layered materials.

[1] http://math.nist.gov/oommf/

[2] http://www.scriptics.com/

[3] http://math.nist.gov/oommf/software.html

If you want to receive email notification of updates to this project, register your email address with the "μMag Announcement" mailing list:

http://www.ctcms.nist.gov/~rdm/email-list.html.

The OOMMF developers are always interested in your comments about OOMMF. See the Credits (Sec. 20) for instructions on how to contact them.

2 Installation

2.1 Requirements

OOMMF software is written in C++ and Tcl. It uses the Tcl-based Tk Windowing Toolkit to create graphical user interfaces that are portable to many varieties of Unix as well as Microsoft Windows 95/98/NT.

Tcl and Tk must be installed before installing OOMMF. Tcl and Tk are available for free download from Scriptics Corporation[4]. We recommend the latest stable versions of Tcl and Tk concurrent with this release of OOMMF. OOMMF requires at least Tcl version 7.5 and Tk version 4.1 on Unix platforms, and requires at least Tcl version 7.6 and Tk version 4.2 on Microsoft Windows platforms. OOMMF software does not support any alpha or beta versions of Tcl/Tk, and each release of OOMMF may not work with later releases of Tcl/Tk. Check the release dates of both OOMMF and Tcl/Tk to ensure compatibility.

A Tcl/Tk installation includes two shell programs. The names of these programs may vary depending on the Tcl/Tk version and the type of platform. The first shell program contains an interpreter for the base Tcl language. In the OOMMF documentation we refer to this program as `tclsh`. The second shell program contains an interpreter for the base Tcl language extended by the Tcl commands supplied by the Tk toolkit. In the OOMMF documentation we refer to this program as `wish`. Consult your Tcl/Tk documentation to determine the actual names of these programs on your platform (for example, `tclsh80.exe` or `wish4.2`).

OOMMF applications communicate via TCP/IP network sockets. This means that OOMMF requires support for networking, even on a stand-alone machine. At a minimum, OOMMF must be able to access the loopback interface so that the host can talk to itself using TCP/IP.

The OOMMF source distribution unpacks into a directory tree containing about 400 files and directories, occupying between 5 and 6 MB of storage. Compiling and linking for each platform consumes approximately an additional 5 MB of storage. The OOMMF distribution containing Windows executables unpacks into a directory tree occupying about 8 MB of storage. **Note:** On a non-compressed `FAT16` file system on a large disk, OOMMF may take up much more disk space. This is because on such systems, the minimum size of any file is large, as much as 32 KB. Since this is much larger than many files in the OOMMF distribution require, a great deal of disk space is wasted.

To build OOMMF software from source code, you will need a C++ compiler capable of handling C++ templates, as well as other software development utilities for your platform.

[4]http://www.scriptics.com/

We do development and test builds on the following platforms, although porting to others should not be difficult:

Platform	Compilers
AIX	Gnu gcc
Alpha/Digital UNIX	Gnu gcc
Alpha/Linux	Gnu gcc/egcs
Alpha/Windows NT	Microsoft Visual C++
HP-UX	HP C++ (cfront)
Intel/Linux	Gnu gcc/egcs
Intel/Windows NT, 95, 98	Microsoft Visual C++, Cygwin gcc
MIPS/IRIX 6 (SGI)	MIPSpro C++
SPARC/Solaris	Sun Workshop C++, Gnu gcc
SPARC/SunOS 4	Sun Workshop C++, Gnu gcc

2.2 Basic Installation

Follow the instructions in the following sections, in order, to prepare OOMMF software for use on your computer.

2.2.1 Download

The latest release of the OOMMF software may be retrieved from the OOMMF download page[5]. Each release is available in two formats. The first format is a gzipped tar file containing an archive of all the OOMMF source code. The second format is a `.zip` compressed archive containing source code and pre-compiled executables for Microsoft Windows 95/98/NT running on an x86-based microprocessor system and compatible with Tcl/Tk version 8.0. Other release formats, e.g., pre-compiled executables for Microsoft Windows NT running on a Digital Alpha Systems RISC-based microprocessor system, and/or compatible with Tcl/Tk version 7.6/4.2 may be made available on request.

For the first format, unpack the distribution archive using gunzip and tar:

```
gunzip -c oommf11a1_19990810.tar.gz | tar xvf -
```

For the other format(s), you will need a utility program to unpack the `.zip` archive. This program must preserve the directory structure of the files in the archive, and it must be able to generate files with names not limited to the traditional MSDOS 8.3 format. Some very

[5]http://math.nist.gov/oommf/software.html

old versions of the pkzip utility do not have these properties. One utility program which is known to work is UnZip[6]. Using your utility, unpack the `.zip` archive, e.g.

```
unzip oommf11a1_19990810.zip
```

For either distribution format, the unpacking sequence creates a subdirectory `oommf` which contains all the files and directories of the OOMMF distribution. If a subdirectory named `oommf` already existed (say, from an earlier OOMMF release), then files in the new distribution overwrite those of the same name already on the disk. Some care may be needed in that circumstance to be sure that the resulting mix of files from an old and a new OOMMF distribution combine to create a working set of files.

2.2.2 Check Your Platform Configuration

After downloading and unpacking the OOMMF software distribution, all the OOMMF software is contained in a subdirectory named `oommf`. Start a command line interface (a shell on Unix, or the MS-DOS Prompt on Microsoft Windows), and change the working directory to the directory `oommf`. Find the Tcl shell program installed as part of your Tcl/Tk installation. In this manual we call the Tcl shell program `tclsh`, but the actual name of the executable depends on the release of Tcl/Tk and your platform type. Consult your Tcl/Tk documentation.

In the root directory of the OOMMF distribution is a file named `platform.tcl`. It will print a summary of your platform configuration when it is evaluated by `tclsh`. This summary describes your platform type, your C++ compiler, and your Tcl/Tk installation. As an example, here is the typical output from `platform.tcl` on a Linux/Alpha system:

```
$ tclsh platform.tcl
<24537> platform   info:
OOMMF release 1.1a1
Platform Name:          linalp
C++ compiler:           /usr/bin/gcc
Tcl configuration file: /usr/local/lib/tclConfig.sh
tclsh:                  /usr/local/bin/tclsh8.0
Tcl release:            8.0.5 (config)  8.0.5 (running)
Tk configuration file:  /usr/local/lib/tkConfig.sh
wish:                   /usr/local/bin/wish8.0
Tk release:             8.0.5 (config)  8.0.5 (running)
```

[6]http://www.cdrom.com/pub/infozip/UnZip.html

If `platform.tcl` doesn't print a summary like that, it should instead print an error message describing why it can't. For example, if your Tcl installation is older than release 7.5, the error message will report that fact. Follow whatever instructions are provided to get `platform.tcl` to print a summary of platform configuration information.

The first line of the example summary reports that OOMMF recognizes the platform by the name `linalp`. OOMMF software recognizes many of the most popular computing platforms, and assigns each a platform name. The platform name is used by OOMMF in index and configuration files and to name directories so that a single OOMMF installation can support multiple platform types. If `platform.tcl` reports the platform name to be "unknown", then you will need to add some configuration files to help OOMMF assign a name to your platform type, and associate with that name some of the key features of your computer. See the section on "Managing OOMMF platform names" (Sec. 2.3.4) for further instructions.

The second line reports what C++ compiler will be used to build OOMMF from its C++ source code. If you downloaded an OOMMF release with pre-compiled binaries for your platform, you may ignore this line. Otherwise, if this line reports "none selected", or if it reports a compiler other than the one you wish to use, then you will need to tell OOMMF what compiler to use. The compiler selection for the example above is recorded in the file `config/cache/linalp.tcl`. To change the compiler selection, that file must be edited. Editing instructions are contained within the file. Of course, on other platforms the name `linalp` in `linalp.tcl` should be replaced with the platform name OOMMF reports for your platform. For example, on a Windows machine using an x86 processor, the corresponding configuration file is `wintel.tcl`.

The next three lines describe the Tcl configuration OOMMF finds on your platform. The first line reports the name of the configuration file installed as part of Tcl, if any. Conventional Tcl installations on Unix systems and within the Cygwin environment on Windows have such a file, usually named `tclConfig.sh`. The Tcl configuration file records details about how Tcl was built and where it was installed. On Windows platforms, this information is recorded in the Windows registry, so it is normal to have `platform.tcl` report "none found". If `platform.tcl` reports "none found", but you know that an appropriate Tcl configuration file is present on your system, you can tell OOMMF where to find the file by setting the environment variable OOMMF_TCL_CONFIG to its absolute location. (For information about setting environment variables, see your operating system documentation.) In unusual circumstances, OOMMF may find a Tcl configuration file which doesn't correctly describe your Tcl installation. In that case, use the environment variable OOMMF_TCL_CONFIG to instruct OOMMF to use a different file that you specify, and edit that file to include a correct description of your Tcl installation.

The second line describing your Tcl installation reports the absolute pathname of the tclsh program. If this differs from the tclsh you used to evaluate platform.tcl, there may be something wrong with your Tcl configuration file. Note that the same tclsh program might be known by several absolute pathnames if there are symbolic links in your Tcl installation. If platform.tcl reports that it cannot find a tclsh program, yet you know where an appropriate one is installed on your system, you can tell OOMMF where to find the tclsh program by setting the environment variable OOMMF_TCLSH to its absolute location.

The third line describing your Tcl installation reports its release number according to two sources. First is the release number recorded in the Tcl configuration file. Second is the release number of the tclsh program used to evaluate platform.tcl. If these numbers do not match, it may indicate something is wrong with your Tcl configuration file. If you have multiple releases of Tcl installed under a common root directory on your computer, there can be only one Tcl configuration file. It is important that you use the Tcl release that corresponds to the Tcl configuration file.

The next three lines describe the Tk configuration OOMMF finds on your platform. They are analogous to the three lines describing the Tcl configuration. The environment variables OOMMF_TK_CONFIG and OOMMF_WISH may be used to tell OOMMF where to find the Tk configuration file and the wish program, respectively.

If platform.tcl indicates problems with your Tcl/Tk installation, it may be easiest to re-install Tcl/Tk taking care to perform a conventional installation. OOMMF deals best with conventional Tcl/Tk installations. If you do not have the power to re-install an existing broken Tcl/Tk installation (perhaps you are not the sysadmin of your machine), you might still install your own copy of Tcl/Tk in your own user space. In that case, if your private Tcl/Tk installation makes use of shared libraries, take care that you do whatever is necessary on your platform to be sure that your private tclsh and wish find and use your private shared libraries instead of those from the system Tcl/Tk installation. This might involve setting an environment variable (such as LD_LIBRARY_PATH). If you use a private Tcl/Tk installation, you also want to be sure that there are no environment variables like TCL_LIBRARY or TK_LIBRARY that still refer to the system Tcl/Tk installation.

Other Configuration Issues If you plan to compile and link OOMMF software from source code, be sure the C++ compiler reported by platform.tcl is properly configured. In particular, the Microsoft Visual C++ command line compiler, cl.exe, may require the running of vcvars32.bat to set up the path and some environment variables. This file is distributed as part of Visual C++. See your compiler documentation for details.

A few other configurations should be checked on Windows platforms. First, OOMMF software needs networking support that recognizes the host name localhost. It may be necessary to edit a file which records that `localhost` is a synonym for the loopback interface (127.0.0.1). If a file named `hosts` exists in your system area (for example, C:\Windows\hosts), be sure it includes an entry mapping 127.0.0.1 to `localhost`. If no `hosts` file exists, but a `hosts.sam` file exists, make a copy of `hosts.sam` with the name `hosts`, and edit the copy to have the `localhost` entry.

Second, in recent releases of Tcl/Tk (version 8.0.3 and later) the directory which holds the `tclsh` and `wish` programs also holds several `*.dll` files that OOMMF software needs to find to run properly. Normally when the OOMMF bootstrap application (Sec. 5) or **mmLaunch** (Sec. 6) is used to launch OOMMF programs, they take care of making sure the necessary `*.dll` files can be found. As an additional measure, you might want to add the directory which holds the `tclsh` and `wish` programs to the list of directories stored in the PATH environment variable. All the directories in the PATH are searched for `*.dll` files needed when starting an executable.

2.2.3 Compiling and Linking

If you downloaded a distribution with pre-compiled executables, you may skip this section.

The compiling and linking of the C++ portions of OOMMF software are guided by the application `pimake` ("Platform Independent Make") which is distributed as part of the OOMMF release. `pimake` is similar in operation to the Unix utility program `make`, but it is written entirely in Tcl so that it will run anywhere Tcl is installed. Just as `make` is controlled by rules in files named `Makefile` or `makefile`, `pimake` is controlled by rules in files named `makerules.tcl`. `pimake` takes an optional argument on its command line which names the target to build. Like all OOMMF applications, pimake is to be executed using the OOMMF bootstrap application (Sec. 5).

If you unpacked the new OOMMF release into directory `oommf` which contained an earlier OOMMF release, build the target `upgrade` with `pimake` to clear away old executables and object files left behind from the compilation of the previous distribution, and to bring all files up to date with the new release:

```
tclsh oommf.tcl -fg pimake upgrade
```

To build all the OOMMF software, run `pimake` in the root directory of the OOMMF distribution.

```
cd .../path/to/oommf
tclsh oommf.tcl -fg pimake
```

Note that on some platforms, you cannot successfully compile OOMMF software if there are OOMMF programs running. Check that all OOMMF programs have terminated (including those in the background) before trying to compile and link OOMMF.

When `pimake` calls on a compiler or other software development utility, the command line is printed, so that you may monitor the build process. Assuming a proper configuration for your platform, `pimake` should be able to compile and link all the OOMMF software without error. If `pimake` reports errors, please send both the *complete* output from `pimake` and the output from `platform.tcl` to the OOMMF developers.

2.2.4 Installing

The current OOMMF release does not support an installation procedure. For now, simply run the executables from the directories in which they were unpacked/built.

2.2.5 Using OOMMF Software

To start using OOMMF software, run the OOMMF bootstrap application (Sec. 5). This may be launched from the command line interface:

```
tclsh oommf.tcl
```

If you prefer, you may launch the OOMMF bootstrap application `oommf.tcl` using whatever graphical "point and click" interface your operating system provides. By default, the OOMMF bootstrap application will start up a copy of the OOMMF application **mmLaunch** (Sec. 6) in a new window.

2.2.6 Reporting Problems

If you encounter problems when installing or using OOMMF, please report them to the OOMMF developers. See Troubleshooting (Sec. 18) for detailed instructions.

2.3 Advanced Installation

The following sections provide instructions for some additional installation options.

2.3.1 Reducing Disk Space Usage

To delete the intermediate files created when building the OOMMF software from source code, use `pimake` to build the target `objclean` in the root directory of the OOMMF distribution.

```
tclsh oommf.tcl -fg pimake objclean
```

Running the `strip` utility on the OOMMF executable files should also reduce their size somewhat.

2.3.2 Other `pimake` targets

The **pimake** application guides the building of OOMMF software according to the rules recorded in `makerules.tcl` files. There are several targets which may be used as arguments to **pimake** to achieve different tasks. When no target is supplied to **pimake** on the command line, the first target in the local `makerules.tcl` file is built, which is usually the target `all`. Each target builds in the current directory and all subdirectories. The conventional targets are:

- **all** Creates all files created by the `configure` target (see below). Compiles and links all the OOMMF executables and libraries. Constructs all index files.
- **configure** Creates subdirectories with the same name as the platform type. Constructs a `port.h` file which includes C++ header information specific to the platform.
- **objclean** Removes the intermediate object files created by the compile and link steps. Leaves working executables in place. Leaves OOMMF in the state of its distribution with pre-compiled executables.
- **clean** Removes the files removed by the `objclean` target. Also removes the executables and libraries created by the `all` target. Leaves the files generated by the `configure` target.
- **distclean** Removes the files removed by the `clean` target. Also removes all files and directories generated by `configure` target. Leaves only the files which are part of the source code distribution.

2.3.3 Local Customizations

OOMMF software supports local customization of some of its features. All OOMMF programs load the file `config/options.tcl`, which contains customization commands as well as editing instructions. As it is distributed, `config/options.tcl` directs those programs that load it to also load the file `config/local/options.tcl`, if it exists. Because future OOMMF releases may overwrite the file `config/options.tcl`, permanent customizations should be made by copying `config/options.tcl` to `config/local/options.tcl` and editing the copy. It is recommended that you leave in the file `config/local/options.tcl` only the customization commands necessary to change those options you wish to modify. Remove all other options so that overwrites by subsequent OOMMF releases are allowed to change the default behavior.

Notable available customizations include the choice of which network port the host service directory application (Sec. 4) uses, and the choice of what program is used for the display of help documentation. By default, OOMMF software uses the application **mmHelp** (Sec. 14), which is included in the OOMMF release, but the help documentation files are standard HTML, so any web browser (for example, Netscape Navigator or Microsoft Internet Explorer) may be used instead. Complete instructions are in the file `config/options.tcl`.

2.3.4 Managing OOMMF Platform Names

OOMMF software classifies computing platforms into different types using the scripts in the directory `config/names` relative to the root directory of the OOMMF distribution. Each type of computing platform is assigned a unique name. These names are used as directory names and in index and configuration files so that a single OOMMF installation may contain platform-dependent sections for many different types of computing platforms.

To learn what name OOMMF software uses to refer to your computing platform, run the Tcl script `platform.tcl` in the OOMMF root directory.

Changing the name OOMMF assigns to your platform First, use `pimake` to build the target `distclean` to clear away any compiled executables built using the old platform name.

```
tclsh oommf.tcl -fg pimake distclean
```

Then, to change the name OOMMF software uses to describe your platform from `foo` to `bar`, simply rename the file

```
    config/names/foo.tcl    to    config/names/bar.tcl
```

and

```
    config/cache/foo.tcl    to    config/cache/bar.tcl.
```

After renaming your platform type, you should recompile your executables using the new platform name.

Adding a new platform type If the script `platform.tcl` reports the platform name `unknown`, then none of the scripts in `config/names/` recognizes your platform type. As an example, to add the platform name `foo` to OOMMF's vocabulary of platform names, create the file `config/names/foo.tcl`. The simplest way to proceed is to copy an existing file in the directory `config/names` and edit it to recognize your platform.

The files in `config/names` include Tcl code like this:

```
Oc_Config New _ \
  [string tolower [file rootname [file tail [info script]]]] {
    # In this block place the body of a Tcl proc which returns 1
    # if the machine on which the proc is executed is of the
    # platform type identified by this file, and which returns 0
    # otherwise.
    #
    # The usual Tcl language mechanism for discovering details
    # about the machine on which the proc is running is to
    # consult the global Tcl variable 'tcl_platform'.  See the
    # existing files for examples, or contact the OOMMF
    # developers for further assistance.
}
```

After creating the new platform name file `config/names/foo.tcl`, you also need to create a new platform cache file `config/cache/foo.tcl`. A reasonable starting point is to copy the file `config/cache/unknown.tcl` for editing. Contact the OOMMF developers for assistance.

Please consider contributing your new platform recognition and configuration files to the OOMMF developers for inclusion in future releases of OOMMF software.

Resolving platform name conflicts If the script `platform.tcl` reports "Multiple platform names are compatible with your computer", then there are multiple files in the directory `config/names/` that return 1 when run on your computer. For each compatible platform name reported, edit the corresponding file in `config/names/` so that only one of them returns 1. Experimenting using `tclsh` to probe the Tcl variable `tcl_platform` should assist you in this task. If that fails, you can explicitly assign a platform type corresponding to your computing platform by matching its hostname. For example, if your machine's host name is `foo.bar.net`:

```
Oc_Config New _ \
  [string tolower [file rootname [file tail [info script]]]] {
    if {[string match foo.bar.net [info hostname]]} {
        return 1
    }
    # Continue with other tests...
}
```

Contact the OOMMF developers if you need further assistance.

2.3.5 Microsoft Windows Options

This section lists installation options for Microsoft Windows.

Adding an OOMMF shortcut to your desktop Right mouse click on the desktop to bring up the configuration dialog, and select **New|Shortcut**. Enter the command line necessary to bring up OOMMF, e.g.,

```
tclsh80 c:\oommf\oommf.tcl
```

Click **Next**> and enter `OOMMF` for the shortcut name. Select **Finish**.

At this point the shortcut will appear on your desktop with either the tclsh or wish icons. Right mouse click on the icon and select **Properties**. Select the **ShortCut** tab, and bring up **Change Icon...** Under **File Name:** enter the OOMMF icon file, e.g.,

```
C:\oommf\oommf.ico
```

Click **OK**. Back on the **Shortcut** tab, change the **Run:** selection to Minimized. Click **OK** to exit the Properties dialog box. Double clicking on the OOMMF icon should now bring up the OOMMF application **mmLaunch**.

Using the Cygwin toolkit The Cygwin Project[7] is a free port of the GNU development environment to Windows NT, 95, and 98, which includes the GNU C++ compiler gcc and a port of Tcl/Tk. OOMMF has been tested against the Beta 20.1 release of Cygwin, and sample `config/names/cygtel.tcl` and `config/cache/cygtel.tcl` files are included in the OOMMF distribution. Use the `cygtclsh80.exe` program as your `tclsh` program when configuring, building, and launching OOMMF software.

Note that OOMMF software determines whether it is running with the Cygwin versions of Tcl/Tk by examining the environment variable TERM. If TERM is set to the value `cygwin`, the Cygwin environment is assumed. If you are using the Cygwin environment, and you set the TERM environment variable to something else, OOMMF software will be confused.

Setting the TCL_LIBRARY environment variable If you encounter difficulties during OOMMF start up, you may need to set the environment variable TCL_LIBRARY.

[7]http://sourceware.cygnus.com/cygwin/

On Windows NT Bring up the Control Panel (e.g., by selecting **Settings|Control Panel** off the Start menu), and select **System**. Go to the **Environment** tab, and enter TCL_LIBRARY as the Variable, and the name of the directory containing `init.tcl` for the Value, e.g.,

```
%SystemDrive%\Program Files\Tcl\lib\tcl8.0
```

Click **Set** and **OK** to finish.

On Windows 95 Edit the file `autoexec.bat`. Add a line such as the following:

```
set TCL_LIBRARY=C:\Program Files\Tcl\lib\tcl8.0
```

Checking .tcl file association on Windows NT As part of the Tcl/Tk installation, files with the `.tcl` extension are normally associated with the `wish` application. This allows Tcl scripts to be launched from Windows Explorer by double-clicking on their icon, or from the NT command line without specifying the `tclsh` or `wish` shells. If this is not working, you may check your installation from the NT command line as follows. First, run the command `assoc .tcl`. This should return the file type associated with the `.tcl` extension, e.g., `TclScript`. Next, use the `ftype` command to check the command line associated with that file type, e.g.,

```
C:\> ftype TclScript
 "C:\Program Files\Tcl\bin\wish80.exe" "%1" %2 %3 %4 %5 %6 %7 %8 %9
```

Note that the quotes are required as shown to protect spaces in pathnames.

3 Quick Start: Example OOMMF Session

STEP 1: Start up the mmLaunch window.

- At the command prompt, when you are in the OOMMF root directory, type

  ```
  tclsh oommf.tcl
  ```

  (The name of the Tcl shell, rendered here as `tclsh`, may vary between systems. This matter is discussed in Sec. 2.1.) Alternatively, you may launch `oommf.tcl` using whatever "point and click" interface is provided by your operating system.
- This will bring up a small window labeled **mmLaunch**. It will come up in background mode, so you will get another prompt in your original window, even before the **mmLaunch** window appears.

STEP 2: Gain access to other useful windows.

- On **mmLaunch** window, check the **localhost** box, causing a menu of user account boxes to appear. Then check the box corresponding to the account you want to compute on. This gives a menu of options:
  - **mmProbEd:** to grab/modify a problem
  - **mmSolve2D:** to control the solver
  - **mmDisp:** to display vector fields
  - **mmGraph:** to form x-y plots
  - **mmDataTable:** to display current values of variables
  - **mmArchive:** to auto-save vector field data (primitive)
- Click on **mmDisp**, **mmGraph**, and/or **mmDataTable**, depending on what form of output you want.

STEP 3: Load a problem.

- On **mmLaunch** window, click on the **mmProbEd** button.
- On **mmProbEd** window, make menu selection **File|Open...** A **Load MIF File** dialog window will appear.
- On this window:
  - Double click in the **Path** subwindow to change directories. Several sample problems can be found in the directory `oommf/app/mmpe/examples`.

  - To load a problem, double click on a `*.mif` file (e.g., prob1.mif) from the list above the **Filter:** subwindow.
  - Modify the problem as desired by clicking on buttons from the main **mmProbEd** window (e.g., **Material Parameters**), and fill out the pop-up forms. A completely new problem may be defined this way.

STEP 4: Initialize the solver.

- On **mmLaunch** window, click on the **mmSolve2D** button to launch an instance of the program **mmSolve2D**.
- Wait for the new solver instance to appear in the **Threads** column in the **mmLaunch** window.
- Check the box next to the **mmSolve2D** entry in the **Threads** column. A window containing an **mmSolve2D** interface will appear.
- On **mmSolve2D** window:
  - Check **Problem Description** under **Inputs**.
  - Check **mmProbEd** under **Source Threads**.
  - Click **LoadProblem**.
  - Wait for more buttons to appear.
  - Check **Scheduled Outputs**.
  - For each desired output (**TotalField**, **Magnetization**, and/or **DataTable**), specify the frequency of update:
    - ∗ Check desired output. This will exhibit the possible output destinations in **Destination Threads**.
    - ∗ Check the box next to the desired Destination Thread. This will exhibit **Schedule** options.
    - ∗ Choose a schedule:
      - · **Iteration:** fill in number and check the box.
      - · **ControlPoint:** fill in number and check the box.
      - · **Interactive:** whenever you click corresponding Interactive output button.

STEP 5: Start the calculation.

- On the **mmSolve2D** window, start the calculation with **Run** or **Relax**.

- If you requested mmDataTable output, check the boxes for the desired quantities on the **mmDataTable** window under the Data menu, so that they appear and are updated as requested in your schedule.
- Similarly, check the box for the desired X-Axis and Y-Axis variables on the **mmGraph** window(s) under the **X-Axis** and **Y-Axis** menus.

STEP 6: Saving results.

- Vector field data (magnetization and effective field) may be interactively written to disk using **mmDisp**, or may be automatically saved via scheduled output to **mmArchive**. For example, to save the magnetization state at each control point, start up an instance of **mmArchive** and select the **ControlPoint** check box for **mmArchive** on the **Magnetization** schedule in the solver. This may be done before starting the calculation. (Control points are points in the simulation where the applied field is stepped. These are typically equilibrium states, but depending on the input `*.mif` file, may be triggered by elapsed simulation time or iteration count.)
- **DataTable** data may be saved using **mmGraph**. Schedule output from the solver to **mmGraph** as desired, and use either the interactive or automated save functionality of **mmGraph** (Sec. 11). You can setup the solver data scheduling before the calculation is started, but must wait for the first data point to configure **mmGraph** before saving any data. As a workaround, you may configure **mmGraph** by sending it the initial solver state interactively, and then use the **Options|clear Data** menu item in **mmGraph** to remove the initializing data point. Alternatively, you may send scheduled output from the solver to **mmArchive**, which will automatically save all the data it receives.

STEP 7: Perform midcourse controls as desired.

- On the **mmSolve2D** window, buttons can stop and restart the calculation:
  - **Reset:** Return to beginning of problem.
  - **LoadProblem:** Restart with a new problem.
  - **Run:** Apply a sequence of fields until all complete.
  - **Relax:** Run the ODE at the current applied field until the next control point is reached.
  - **Pause:** Click anytime to stop the solver. Restart with **Run** or **Relax**.
  - **Field-:** Apply the previous field again.
  - **Field+:** Apply the next field in the list.
- Output options can be changed and new output windows opened.

STEP 8: Exit OOMMF.

- On the **mmSolve2D** window, terminate the simulation with **Exit**.
- Terminate each **mmArchive** instance by hitting the **Exit** button in its user interface window.
- Use the **File|Exit** menu on each remaining window to exit.

4 OOMMF Architecture Overview

Before describing each of the applications which comprise the OOMMF software, it is helpful to understand how these applications work together. OOMMF is not structured as a single program. Instead it is a collection of programs, each specializing in some task needed as part of a micromagnetic simulation system. An advantage of this modular architecture is that each program may be improved or even replaced without a need to redesign the entire system. Because the state of the art in micromagnetic simulation is continuing to evolve, this flexibility is essential for the longevity of a micromagnetic simulation system.

The OOMMF programs work together by providing services to one another. The programs communicate over Internet (TCP/IP) connections, even when the programs are running on a common host. An advantage of this design is that distributed operation of OOMMF programs over a networked collection of hosts is supported in the basic design, and will be available in a future release.

When two OOMMF applications are in the relationship that one is requesting a service from the other, it is convenient to introduce some clarifying terminology. Let us refer to the application that is providing a service as the "server application" and the application requesting the service as the "client application." Note that a single application can be both a server application in one service relationship and a client application in another service relationship.

Each server application provides its services on a particular Internet port, and needs to inform potential client applications how to obtain its service. Each client application needs to be able to look up possible providers of the service it needs. The intermediary which brings server applications and client applications together is another application called the "account service directory." There may be at most one account service directory application running under the user ID of each user account on a host. Each account service directory keeps track of all the services provided by OOMMF server applications running under its user account on its host and the corresponding Internet ports at which those services may be obtained. OOMMF server applications register their services with the corresponding account service directory application. OOMMF client applications look up service providers running under a particular user ID in the corresponding account server directory application.

The account service directory applications simplify the problem of matching servers and clients, but they do not completely solve it. OOMMF applications still need a mechanism to find out how to obtain the service of the account service directory applications! Another application, called the "host service directory" serves this function. Only one copy of the host service directory application runs on each host. Its sole purpose is to tell OOMMF applications where to obtain the services of account service directories on that host. Because only one copy of this application runs per host, it can provide its service on a well-known port which is configured into the OOMMF software. By default, this is port 15136. OOMMF software can be customized (Sec. 2.3.3) to use a different port number.

The account service directory applications perform another task as well. They launch other programs under the user ID for which they manage service registration. The user controls the launching of programs through the interface provided by the application **mm-Launch** (See Sec. 6), but it is the account service directory application that actually spawns a subprocess for the new application. Because of this architecture, most OOMMF applications are launched as child processes of an account service directory application. These child processes inherit their environment from their parent account service directory application, including their working directory, and other key environment variables, such as `DISPLAY`. Each account service directory application sets its working directory to the root directory of the OOMMF distribution. Future releases of OOMMF software will likely be based on a revised architecture which alleviates these restrictions.

These service directory applications are vitally important to the operation of the total OOMMF micromagnetic simulation system. However, it would be easy to overlook them. They act entirely “behind the scenes” without a user interface window. Furthermore, they are never launched by the user. When any server application needs to register its service, if it finds that these service directory applications are not running, it launches new copies of them. In this way the user can be sure that if any OOMMF server applications are running, then so are the service directory applications needed to direct clients to its service. After all server applications terminate, and there are no longer any services registered with a service directory application, it will terminate after a timeout expires.

In the sections which follow, the OOMMF applications are described in terms of the services they provide and the services they require.

5 Command Line Launching

Some of the OOMMF applications are platform-independent Tcl scripts. Some of them are Tcl scripts that require special platform-dependent interpreters. Others are platform-dependent, compiled C++ applications. It is likely that some of them will change status in later releases of OOMMF. Each of these types of application requires a different command line for launching. Rather than require all OOMMF users to manage this complexity, we provide a pair of programs that provide simplified interfaces for launching OOMMF applications.

The first of these is used to launch OOMMF applications from the command line. Because its function is only to start another program, we refer to this program as the "bootstrap application." The bootstrap application is the Tcl script `oommf.tcl`. In its simplest usage, it takes a single argument on the command line, the name of the application to launch. For example, to launch **mmGraph** (Sec. 11), the command line is:

```
tclsh oommf.tcl mmGraph
```

The search for an application matching the name is case-insensitive. (Here, as elsewhere in this document, the current working directory is assumed to be the OOMMF root directory. For other cases, adjust the pathname as appropriate.) As discussed in Sec. 2.1, the name of the Tcl shell, rendered here as `tclsh`, may vary between systems.

If no command line arguments are passed to the bootstrap application, by default it will launch the application **mmLaunch** (Sec. 6).

Some of the OOMMF applications take optional arguments when launched from the command line. Any command line arguments to the bootstrap application placed after the application name will be passed as command line arguments to the named application. For example, most applications support a `-console` option, which will bring up an additional window providing a command line interface into the application's Tcl interpreter. (The console is mainly used for debugging.)

The bootstrap application also supports some more complicated, more rarely used processing of its command line arguments. By default, the bootstrap application launches the named application in the background and exits. If the first argument to the bootstrap program is the option `-fg`, however, the bootstrap program will block until the application it launches (now named in the second command line argument) exits.

The application to be launched by the bootstrap application is usually requested as a simple name. When a particular version of an application is required, though, `oommf.tcl` allows the user to include that requirement in the command line arguments. For example:

```
tclsh oommf.tcl "mmGraph 1.1"
```

will guarantee that the instance of the application mmGraph it launches is of at least version 1.1. If no copy of mmGraph satisfying the version requirement can be found, an error is reported.

For a summary of all command line options recognized by the bootstrap application, run:

```
tclsh oommf.tcl -h
```

The bootstrap application should be infrequently used by most users. The application **mmLaunch** (Sec. 6) provides a more convenient graphical interface for launching applications. The main uses for the bootstrap application are launching **mmLaunch**, launching **pimake**, launching programs which make up the OOMMF Batch System (Sec. 16) and other programs which are inherently command line driven, and in circumstances where the user wishes to precisely control the command line arguments passed to an OOMMF application or the environment in which an OOMMF application runs.

Platform Issues

The Tcl script `oommf.tcl` begins with the lines:

```
#!/bin/sh
# \
exec tclsh "$0" ${1+"$@"}
```

On most Unix platforms, if `oommf.tcl` is marked executable, the interpreter `tclsh` (on the execution path) will be invoked to interpret the script. If the Tcl shell program cannot be invoked by the name `tclsh` on your computer, edit the first lines of `oommf.tcl` to use the proper name. Better still, use symbolic links or some other means to make the Tcl shell program available by the name `tclsh`. The latter solution will not be undone by file overwrites from OOMMF upgrades.

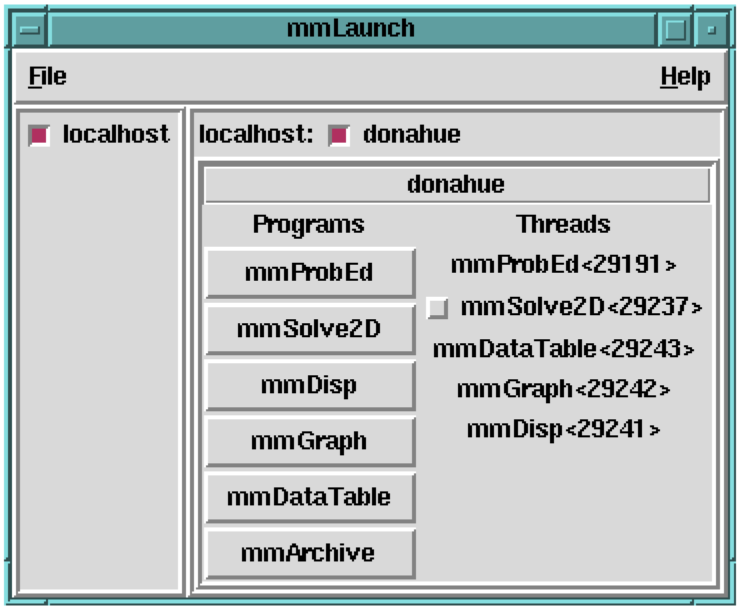
If in addition, the directory `.../path/to/oommf` is in the execution path, the command line can be as simple as:

```
oommf.tcl <appName>
```

from any working directory.

On Windows platforms, because `oommf.tcl` has the file extension `.tcl`, it is normally associated by Windows with the `wish` interpreter. The `oommf.tcl` script has been specially written so that either `tclsh` or `wish` is a suitable interpreter. This means that simply double-clicking on an icon associated with the file `oommf.tcl` (say, in Windows Explorer) will launch the bootstrap application with no arguments. This will result in the default behavior of launching the application **mmLaunch**, which is suitable for launching other OOMMF applications. (If this doesn't work, refer back to the Windows Options section in the installation instructions, Sec. 2.3.5.)

6 OOMMF Launcher/Control Interface: mmLaunch



Overview

The application **mmLaunch** launches, monitors, and controls other OOMMF applications. It is the OOMMF application that is most closely connected to the account service directory and host service directory applications that run behind the scenes. It also provides user interfaces to any applications, notably **mmSolve2D** (Sec. 9), that do not have their own user interface window.

Launching

mmLaunch should be launched using the bootstrap application (Sec. 5). The command line is

```
tclsh oommf.tcl mmLaunch [-console]
```

Controls

Upon startup, **mmLaunch** displays a panel of checkbuttons, one for each host service directory to which it is connected. In the current release of OOMMF there is only one checkbutton—**localhost**. Future releases of **mmLaunch** will be able to connect to remote hosts as well. If there is no host service directory running on the localhost when **mmLaunch** is launched, **mmLaunch** will start one. In that circumstance, there may be some delay before the **localhost** check button appears.

Toggling the **localhost** checkbutton toggles the display of an interface to the host service directory. The host service directory interface consists of a row of checkbuttons, one for each account service directory registered with the host service directory. Each checkbutton is labeled with the user ID of the corresponding account service directory. For most users, there will be only one checkbutton, labeled with the user's own account ID, except on Windows, where the dummy account ID **oommf** is displayed instead. If there is no account service directory running for the account under which **mmLaunch** was launched, **mmLaunch** will start one. In that circumstance, there may be some delay before the account checkbutton appears.

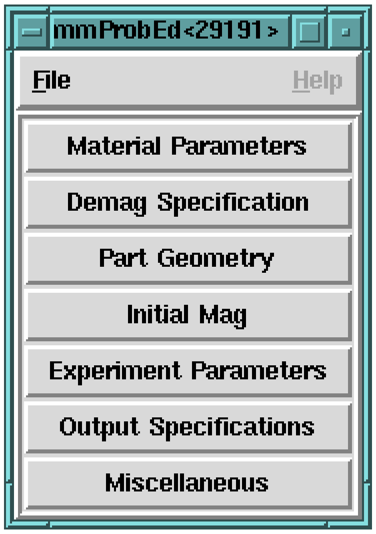
Toggling an account checkbutton toggles the display of an interface to the corresponding account service directory. The account service directory interface consists of two columns. The **Programs** column contains buttons labeled with the names of OOMMF applications that may be launched under the account managed by this account service directory. Clicking on one of these buttons launches the corresponding application. Only one click is needed, though there will be some delay before the launched application displays a window to the user. Multiple clicks will launch multiple copies of the application. Note: The launching is actually handled by the account service directory application (Sec. 4), which sets the initial working directory to the OOMMF root directory.

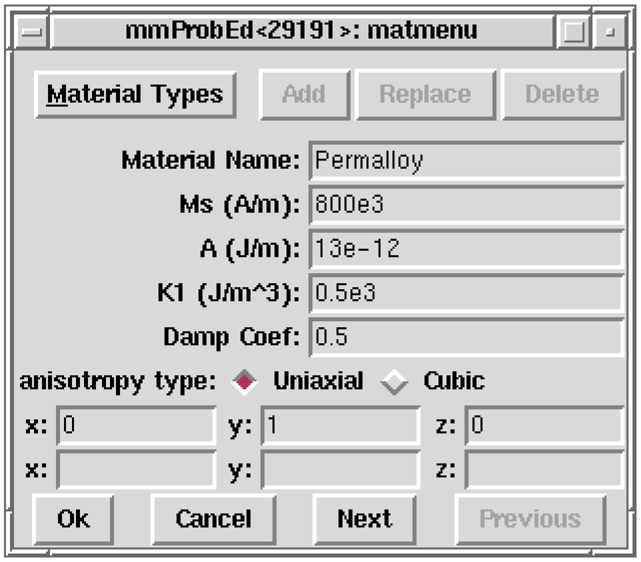
The **Threads** column is a list of all the OOMMF applications currently running under the account that are registered with the account service directory. The list includes both the application name and an ID number by which multiple copies of the same application may be distinguished. This ID number is also displayed in the title bar of the corresponding application's user interface window. When an application exits, its entry is automatically removed from the Threads list.

Any of the running applications that do not provide their own interface window will be displayed in the **Threads** list with a checkbutton. The checkbutton toggles the display of an interface which **mmLaunch** provides on behalf of that application. The only OOMMF applications currently using this service are **mmSolve2D** (Sec. 9), **mmArchive** (Sec. 13), and **batchsolve** (Sec. 16.1). These interfaces are described in the documentation for the corresponding applications.

The menu selection **File|Exit** terminates the **mmLaunch** application. The menu **Help** provides the usual help facilities.

7 Micromagnetic Problem Editor: mmProbEd





Material Name: Permalloy
Ms (A/m): 800e3
A (J/m): 13e-12
K1 (J/m^3): 0.5e3
Damp Coef: 0.5
anisotropy type: [✓ Uniaxial] Cubic
x: 0 y: 1 z: 0
x: y: z:

Overview

The application **mmProbEd** provides a user interface for creating and editing micromagnetic problem descriptions in the *Micromagnetic Input Format* (MIF) (Sec. 17.1). **mmProbEd** also acts as a server, supplying problem descriptions to running micromagnetic solvers.

Launching

mmProbEd may be started either by selecting the **mmProbEd** button on **mmLaunch**, or from the command line via

```
tclsh oommf.tcl mmProbEd [-console]
```

Inputs

The menu selection **File|Open...** displays a dialog box for selecting a file from which to load a MIF problem description. Several example files are included in the OOMMF release in the directory `app/mmpe/examples`. At startup, **mmProbEd** loads the problem contained in `app/mmpe/init.mif` as an initial problem. Note: When loading a file, **mmProbEd** discards comments and records it does not understand. Use the **FileSource** application (Sec.8) to serve unmodified problem descriptions.

Outputs

The menu selection **File|Save as...** displays a dialog box for selecting/entering a file in which the problem description currently held by **mmProbEd** is to be saved. Because the internal data format use by **mmProbEd** is an unordered array that does not include comments (or unrecognized records), the simple operation of reading in a MIF file and then writing it back out may alter the file.

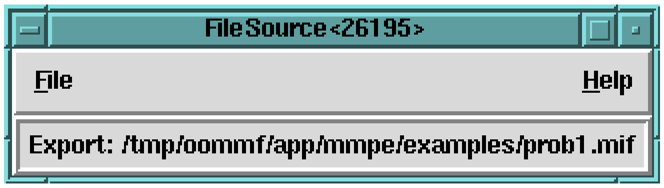
Each instance of **mmProbEd** contains exactly one problem description at a time. Each also services requests from client applications (typically solvers) for the problem description it contains.

Controls

The main panel in the **mmProbEd** window contains buttons corresponding to the sections in a MIF problem description. Selecting a button brings up another window through which the contents of that section of a problem description may be edited. The MIF sections and the elements they contain are described in detail in the MIF (Sec. 17.1) documentation. Only one editing window is displayed at a time. The windows may be navigated in order using their **Next** or **Previous** buttons.

The menu selection **File|Exit** terminates the **mmProbEd** application. The menu **Help** provides the usual help facilities.

8 Micromagnetic Problem File Source: FileSource



Export: /tmp/oommf/app/mmpe/examples/prob1.mif

Overview

The application **FileSource** provides the same service as **mmProbEd** (Sec. 7), supplying a MIF description of a micromagnetic problem to a solver. As the MIF specification evolves, **mmProbEd** may lag behind. There may be new fields in the MIF specification that **mmProbEd** is not capable of editing, or which **mmProbEd** may not pass on to solvers after loading them in from a file. To make use of such fields, a MIF file may need to be edited "by hand" using a general purpose text editor. **FileSource** may then be used to supply the MIF problem description contained in a file to a solver without danger of corrupting its contents.

Launching

FileSource must be launched from the command line. You may specify on the command line the MIF problem description file it should serve to client applications. The command line is

```
tclsh oommf.tcl FileSource [filename] [-console]
```

Although **FileSource** does not appear on the list of **Programs** that **mmLaunch** offers to launch, running copies do appear on the list of **Threads** since they do provide a service registered with the account service directory.

Inputs

FileSource takes its MIF problem description from the file named on the command line, or from a file selected through the **File|Open** dialog box. No checking of the file contents against the MIF specification is performed. The file contents are passed uncritically to any client application requesting a problem description. Those client applications should raise errors when presented with invalid problem descriptions.

Outputs

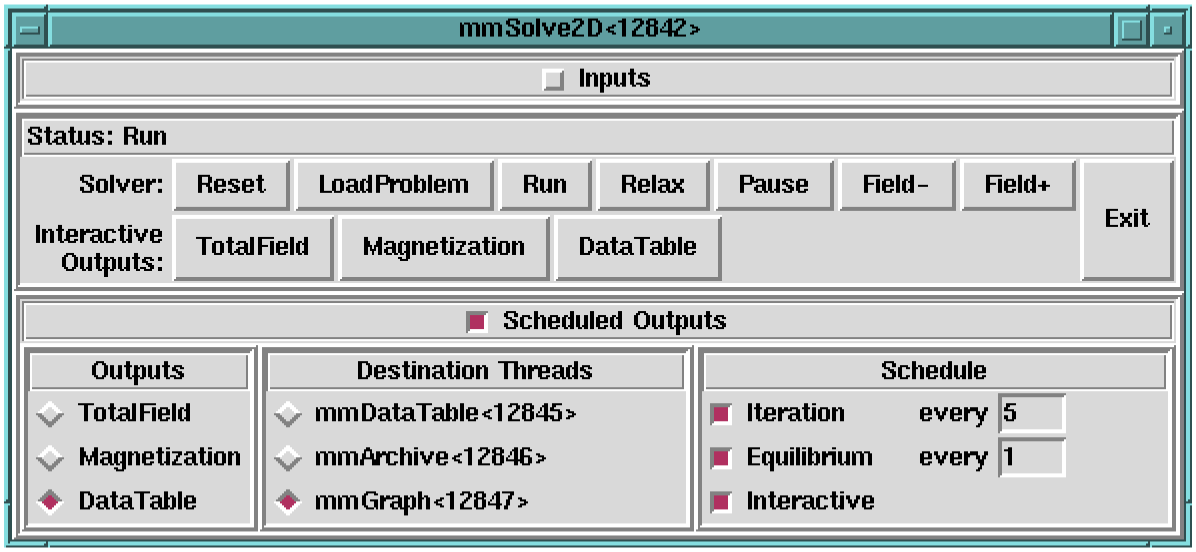
Each instance of **FileSource** provides the contents of exactly one file at a time. The file name is displayed in the **FileSource** window to help the user associate each instance of **FileSource** with the data file it provides. Each instance of **FileSource** accepts and services requests from client applications (typically solvers) for the contents of the file it exports.

The contents of the file are read at the time of the client request, so if the contents of a file change between the time of the **FileSource** file selection and the arrival of a request from a client, the new contents will be served to the client application.

Controls

The menu selection **File | Exit** terminates the **FileSource** application. The **Help** menu provides the usual help facilities.

9 The 2D Micromagnetic Solver: mmSolve2D



Status: Run

Outputs	Destination Threads	Schedule
TotalField	mmDataTable<12845>	✓ Iteration every 5
Magnetization	mmArchive<12846>	✓ Equilibrium every 1
● DataTable	● mmGraph<12847>	✓ Interactive

Overview

The application **mmSolve2D** is a micromagnetic computation engine capable of solving problems defined on two-dimensional square grids of three-dimensional spins. Within the OOMMF architecture (see Sec. 4), **mmSolve2D** is both a server and a client application. **mmSolve2D** is a client of problem description server applications, data table display and storage applications, and vector field display and storage applications. **mmSolve2D** is the server of a solver control service for which the only client is **mmLaunch** (Sec. 6). It is through this service that **mmLaunch** provides a user interface window (shown above) on behalf of **mmSolve2D**.

Launching

mmSolve2D may be started either by selecting the **mmSolve2D** button on mmLaunch, or from the command line via

```
tclsh oommf.tcl mmSolve2D [-console] [-tk <0|1>] [-restart <0|1>]
```

The command line options `-tk 0` and `-tk 1` determine whether or not **mmSolve2D** has its own window. By default it does not (`-tk 0`). With no window, an instance of **mmSolve2D** may keep running while a window system goes down and comes back up again (on some platforms). This is useful since a single instance of **mmSolve2D** may need to run for hours or days at a time to solve one problem. When **mmSolve2D** does have a window (`-tk 1`), it does not place anything in its window, nor does it even display its window. The only effects of the presence of the window are that any error messages will be displayed in pop-up windows, that the `-console` option becomes available for displaying a console window in which interactive Tcl commands may be typed, and that (on some platforms) the termination of the windowing system will terminate **mmSolve2D**. Running **mmSolve2D** with the `-tk 1` option is generally useful only for debugging.

There is another aspect to Tk availability. If a problem is loaded that uses a bitmap mask file (Sec. 17.1), and if that mask file is not in the PPM P3 (text) format, then **mmSolve2D** will launch **any2ppm** (Sec. 15.2) to convert it into the PPM P3 format. Since **any2ppm** requires Tk, at the time the mask file is read a valid display must be available. See the **any2ppm** documentation for details.

Since **mmSolve2D** does not present any user interface window of its own, it depends on **mmLaunch** to provide an interface on its behalf. The entry for an instance of **mmSolve2D** in the **Threads** column of any running copy of **mmLaunch** has a checkbutton next to it. This button toggles the presence of a user interface window through which the user may control that instance of **mmSolve2D**. The user interface window is divided into panels, providing user interfaces to the **Inputs**, **Outputs**, and **Controls** of **mmSolve2D**.

The `-restart` option affects the behavior of the solver when a new problem is loaded. If `-restart` is set to 1, then the solver will look for *`basename`*`.log` and *`basename`*`*.omf` files to restart a previous run from the last saved state (where *`basename`* is the "Base Output Filename" specified in the input MIF problem specification file (Sec. 17.1)). If these files cannot be found, then a warning is issued and the solver falls back to the default behavior (equivalent to `-restart 0`) of starting the problem from scratch. The specified `-restart` setting holds for **all** problems fed to the solver, not just the first. (There is currently no interactive way to change the value of this switch.)

Inputs

The top panel of the user interface window may be opened and closed by toggling the **Inputs** checkbutton. When open, the **Inputs** panel reveals two subpanels. The left subpanel contains a list of the inputs required by **mmSolve2D**. There is only one item in the list: **ProblemDescription**. When **ProblemDescription** is selected, the right subpanel (labeled **Source Threads**) displays a list of applications that can supply a problem description. The user selects from among the listed applications the one from which **mmSolve2D** should request a problem description.

Outputs

When **mmSolve2D** has outputs available to be controlled, a **Scheduled Outputs** checkbutton appears in the user interface window. Toggling the **Scheduled Outputs** checkbutton causes a bottom panel to open and close in the user interface window. When open, the **Scheduled Outputs** panel contains three subpanels. The **Outputs** subpanel is filled with a list of the types of output **mmSolve2D** can generate while solving the loaded problem. The three elements in this list are **TotalField**, for the output of a vector field representing the total effective field, **Magnetization**, for the output of a vector field representing the current magnetization state of the grid of spins, and **DataTable**, for the output of a table of data values describing other quantities of interest calculated by **mmSolve2D**.

Upon selecting one of the output types from the **Outputs** subpanel, a list of applications appears in the **Destination Threads** subpanel which provide a display and/or storage service for the type of output selected. The user may select from this list those applications to which the selected type of output should be sent.

For each application selected, a final interface is displayed in the **Schedule** subpanel. Through this interface the user may set the schedule according to which the selected type of data is sent to the selected application for display or storage. The schedule is described relative to events in **mmSolve2D**. An **Iteration** event occurs at every step in the solution of the ODE. A **ControlPoint** event occurs whenever the solver determines that a control point specification is met. (Control point specs are discussed in the Experiment parameters paragraph in the MIF documentation (Sec. 17.1), and are triggered by solver equilibrium, simulation time, and iteration count conditions.) An **Interactive** event occurs for a particular output type whenever the corresponding "Interactive Outputs" button is clicked in the **Runtime Control** panel. The **Interactive** schedule gives the user the ability to interactively force data to be delivered to selected display and storage applications. For the **Iteration** and **ControlPoint** events, the granularity of the output delivery schedule is under user control. For example, the user may elect to send vector field data describing the current magnetization state to an **mmDisp** instance for display every 25 iterations of the ODE, rather than every iteration.

The quantities included in **DataTable** output produced by **mmSolve2D** include:

- **Iteration:** The iteration count of the ODE solver.
- **Field Updates:** The number of times the ODE solver has calculated the effective field.
- **Sim Time (ns):** The elapsed simulated time.

- **Time Step (ns):** The interval of simulated time spanned by the last step taken in the ODE solver.
- **Step Size:** The magnitude of the last step taken by the ODE solver as a normalized value. (This is currently the time step in seconds, multiplied by the gyromagnetic ratio times the damping coefficient times M_s.)
- **Bx, By, Bz (mT):** The x, y, and z components of the nominal applied field (see Sec. 17.1, Experimental parameters paragraph).
- **B (mT):** The magnitude of the nominal applied field (always non-negative).
- **|m x h|:** The maximum of the point-wise quantity $\|\mathbf{M} \times \mathbf{H}_{\text{eff}}\|/M_s^2$ over all the spins. This "torque" value is used to test convergence to an equilibrium state (and raise control point –torque events).
- **Mx/Ms, My/Ms, Mz/Ms:** The x, y, and z components of the average magnetization of the magnetically active elements of the simulated part.
- **Total Energy (J/m^3):** The total average energy density for the magnetically active elements of the simulated part.
- **Exchange Energy (J/m^3):** The component of the average energy density for the magnetically active elements of the simulated part due to exchange interactions.
- **Anisotropy Energy (J/m^3):** The component of the average energy density for the magnetically active elements of the simulated part due to crystalline anisotropy.
- **Demag Energy (J/m^3):** The component of the average energy density for the magnetically active elements of the simulated part due to self-demagnetizing fields.
- **Zeeman Energy (J/m^3):** The component of average energy density for the magnetically active elements of the simulated part due to interaction with the applied field.
- **Max Angle:** The maximum angle (in degrees) between the magnetization orientation of any pair of neighboring spins in the grid. (The neighborhood of a spin is the same as that defined by the exchange energy calculation.)

In addition, the solver automatically keeps a log file that records the input problem specification and miscellaneous runtime information. The name of this log file is `*basename*.log`, where `*basename*` is the "Base Output Filename" specified in the input problem specification. If this file already exists, then new entries are appended to the end of the file.

Controls

The middle section of the user interface window contains a series of buttons providing user control over the solver. After a problem description server application has been selected, the **LoadProblem** button triggers a fetch of a problem description from the selected server. The **LoadProblem** button may be selected at any time to (re-)load a problem description from the currently selected server. After loading a new problem the solver goes automatically into a paused state. (If no problem description server is selected when the **LoadProblem** button is invoked, nothing will happen.) The **Reset** button operates similarly, except that the current problem specifications are used.

Once a problem is loaded, the solver can be put into any of three states: run, relax and pause. Selecting **Relax** puts the solver into the "relax" state, where it runs until a control point is reached, after which the solver pauses. The **Run** selection differs in that when a control point is reached, the solver automatically steps the nominal applied field to the next value, and continues. In "run" mode the solver will continue to process until there are no more applied field states in the problem description. (Field state schedules are discussed below.) At any time the **Pause** button may be selected to pause the solver. The solver will stay in this state until the user reselects either **Run** or **Relax**. The current state of the solver is indicated in the **Status** line in the center panel of the user interface window.

The problem description (in MIF format) specifies a fixed applied field schedule (see Sec. 17.1, Experimental parameters paragraph). This schedule defines an ordered list of applied fields, which the solver in "run" mode steps through in sequence. The **Field-** and **Field+** buttons allow the user to interactively adjust the applied field sequence. Each click on the **Field+** button advances forward one step through the specified schedule, while **Field-** reverses that process. In general, the step direction is *not* related to the magnitude of the applied field. Also note that hitting these buttons does not generate a "ControlPoint" event. In particular, if you are manually accelerating the progress of the solver through a hysteresis loop, and want to send non-ControlPoint data to a display or archive widget before advancing the field, then you must use the appropriate "Interactive Output" button.

The second row of buttons in the interaction control panel, **TotalField**, **Magnetization** and **DataTable**, allow the user to view the current state of the solver at any time. These buttons cause the solver to send out data of the corresponding type to all applications for which the "Interactive" schedule button for that data type has been selected, as discussed in the Outputs section above.

At the far right of the solver controls is the **Exit** button, which terminates **mmSolve2D**. Simply closing the user interface window does not terminate **mmSolve2D**, but only closes the user interface window. To kill the solver the **Exit** button must be pressed.

Details

Given a problem description, **mmSolve2D** integrates the Landau-Lifshitz equation [5, 7]

$$\frac{d\mathbf{M}}{dt} = -\gamma\,\mathbf{M} \times \mathbf{H}_{\text{eff}} - \frac{\gamma\alpha}{M_s}\,\mathbf{M} \times (\mathbf{M} \times \mathbf{H}_{\text{eff}})\,, \tag{1}$$

where

$\mathbf{M}$	is the pointwise magnetization (A/m),
$\mathbf{H}_{\text{eff}}$	is the pointwise effective field (A/m),
γ	is the gyromagnetic ratio (m/(A·s)),
α	is the damping coefficient (dimensionless).

The effective field is defined as

$$\mathbf{H}_{\text{eff}} = -\mu_0^{-1}\frac{\partial E}{\partial \mathbf{M}}.$$

The average energy density E is a function of $\mathbf{M}$ specified by Brown's equations [3], including crystalline anisotropy, exchange, self-magnetostatic (demagnetization) and applied field (Zeeman) terms.

The micromagnetic problem is impressed upon a regular 2D grid of squares, with 3D magnetization spins positioned at the centers of the cells. Note that the constraint that the grid be composed of square elements takes priority over the requested size of the grid. The actual size of the grid used in the computation will be the nearest integral multiple of the grid's cell size to the requested size. It is important when comparing the results from grids with different cell sizes to account for the possible change in size of the overall grid. At present, Neumann boundary conditions are assumed.

The crystalline anisotropy and applied field energy terms are calculated assuming constant magnetization in each cell. The exchange energy is calculated using the eight-neighbor bilinear interpolation described in [4]. The more common four-neighbor scheme is available as a compile-time option. See the file `app/mmsolve/magelt.cc` for details.

The self-magnetostatic field is calculated using fast Fourier transform (FFT) techniques. Inside each cell, the magnetization is assumed constant along the z-axis. In the xy-plane, two models are supported: constant magnetization and constant charge. The model is selected as part of the problem description in MIF format; for details see Sec. 17.1: Demag specification.

The Landau-Lifshitz ODE (1) is integrated using a second order predictor-corrector technique of the Adams type. The right side of (1) at the current and previous step is extrapolated forward in a linear fashion, and is integrated across the new time interval to obtain a quadratic prediction for $\mathbf{M}$ at the next time step. (At each stage the spins are renormalized to M_s before evaluating the energy and effective fields.) The right side of (1) is evaluated at the predicted $\mathbf{M}$, which is then combined with the value at the current step to produce a linear interpolation of $d\mathbf{M}/dt$ across the new interval. This is then integrated to obtain the final estimate of $\mathbf{M}$ at the new step. The local (one step) error of this procedure should be $O(\Delta t^3)$.

The step is accepted if the total energy of the system decreases, and the maximum error between the predicted and final $\mathbf{M}$ is smaller than a nominal value. If the step is rejected, then the step size is reduced and the integration procedure is repeated. If the step is accepted, then the error between the predicted and final $\mathbf{M}$ is used to adjust the size of the next step. No fixed ratio between the previous and current time step is assumed.

A first order Euler step is used to prime the predictor-corrector solver. A fourth order Runge-Kutta solver is included in the code, and is used as a backup in case the predictor-corrector fails to find a valid step. The Runge-Kutta solver is not selectable as the primary solver at runtime, but may be so selected at compile time by defining the `RUNGE_KUTTA_ODE` macro. See the file `app/mmsolve/grid.cc` for all details of the integration procedure.

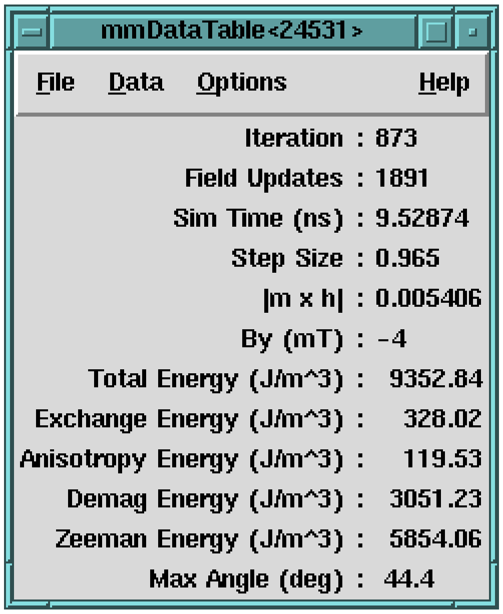
For a given applied field, the integration continues until a control point (cf. Experiment parameters paragraph in Sec. 17.1) is reached. A control point event may be raised by the ODE iteration count, elapsed simulation time, or by the maximum value of $\|\mathbf{M} \times \mathbf{H}_{\text{eff}}\|/M_s^2$ dropping below a specified control point –torque value (implying an equilibrium state has been reached).

Known Bugs

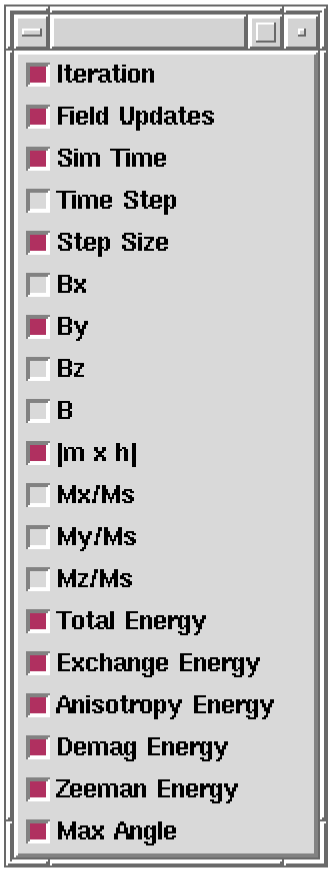
When multiple copies of **mmLaunch** are used, each can have its own interface to a running copy of **mmSolve2D**. When the interface presented by one copy of **mmLaunch** is used to set the output schedule in **mmSolve2D**, those settings are not reflected in the interfaces presented by other copies of **mmLaunch**. For example, although the first interface sets a schedule that DataTable data is to be sent to an instance of **mmGraph** every third Iteration, there is no indication of that schedule presented to the user in the second interface window. It is unusual to have more than one copy of **mmLaunch** running simultaneously. However, this bug also appears when one copy of **mmLaunch** is used to load a problem and start a solver, and later a second copy of **mmLaunch** is used to monitor the status of that running solver.

A bug in the network traffic handling code of Tcl on Windows 95 and Windows 98 systems can sometimes interfere with communications between the control interface of **mmSolve2D** and the actual computation engine. If **mmSolve2D** is sending out data to two or more data display services every iteration, the network traffic used to send out that data can "crowd out" the receipt of control messages from the control interface. You may observe this as a long delay between the time you click the **Pause** button and the time the solver stops iterating. This bug first appeared in Tcl release 8.0.3, and remained through Tcl release 8.1.1. It is expected that this bug will be fixed in Tcl release 8.2, anticipated in August 1999. Until then, a workaround is to install Tcl/Tk 8.0p2 on your Windows 95 or Windows 98 system. Other platforms do not have this problem.

10 Data Table Display: mmDataTable



```
                Iteration : 873
            Field Updates : 1891
            Sim Time (ns) : 9.52874
                Step Size : 0.965
                  |m x h| : 0.005406
                  By (mT) : -4
      Total Energy (J/m^3) :   9352.84
   Exchange Energy (J/m^3) :    328.02
 Anisotropy Energy (J/m^3) :    119.53
      Demag Energy (J/m^3) :   3051.23
    Zeeman Energy (J/m^3) :   5854.06
          Max Angle (deg) :  44.4
```



- ☑ Iteration
- ☑ Field Updates
- ☑ Sim Time
- ☐ Time Step
- ☑ Step Size
- ☐ Bx
- ☑ By
- ☐ Bz
- ☐ B
- ☑ |m x h|
- ☐ Mx/Ms
- ☐ My/Ms
- ☐ Mz/Ms
- ☑ Total Energy
- ☑ Exchange Energy
- ☑ Anisotropy Energy
- ☑ Demag Energy
- ☑ Zeeman Energy
- ☑ Max Angle

Overview

The application **mmDataTable** provides a data display service to its client applications. It accepts data from clients and displays it in a window. Its typical use is to display the evolving values of quantities computed by a micromagnetic solver program.

Launching

mmDataTable may be started either by selecting the **mmDataTable** button on mm-Launch, or from the command line via

```
tclsh oommf.tcl mmDataTable [-console]
```

Inputs

The client application(s) that send data to **mmDataTable** for display control the flow of data. The user, interacting with the **mmDataTable** window, controls how the data is displayed. Upon launch, **mmDataTable** displays only a menubar. Upon user request, a display window below the menubar displays data values.

Each message from a client contains a list of (name, value, units) triples containing data for display. For example, one element in the list might be {`Magnetization 800000 A/m`}. **mmDataTable** stores the latest value it receives for each name. Earlier values are discarded when new data arrives from a client.

Outputs

mmDataTable does not support any data output or storage facilities. To save tabular data, use the **mmGraph** (Sec. 11) or **mmArchive** (Sec. 13) applications.

Controls

The **Data** menu holds a list of all the data names for which **mmDataTable** has received data. Initially, **mmDataTable** has received no data from any clients, so this menu is empty. As data arrives from clients, the menu fills with the list of data names. Each data name on the list lies next to a checkbutton. When the checkbutton is toggled from off to on, the corresponding data name and its value and units are displayed at the bottom of the display window. When the checkbutton is toggled from on to off, the corresponding data name is removed from the display window. In this way, the user selects from all the data received what is to be displayed. Selecting the dashed rule at the top of the **Data** menu detaches it so the user may easily click multiple checkbuttons.

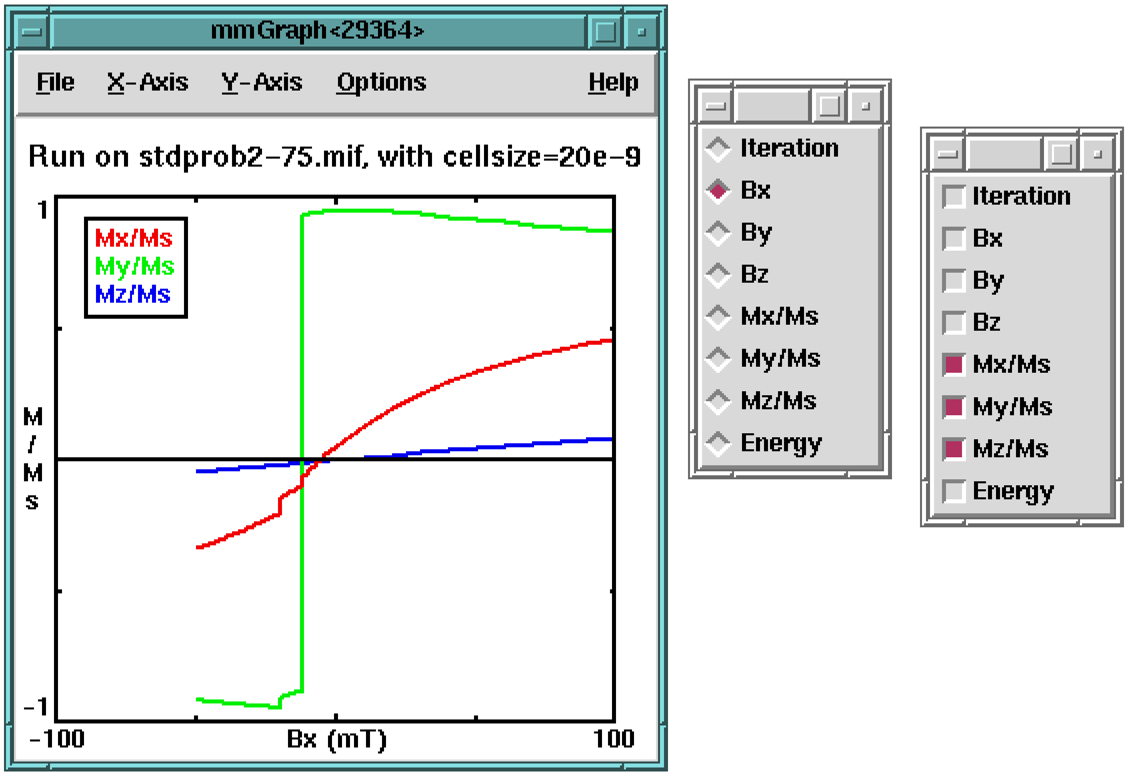
Displayed data values can be individually selected (or deselected) with a left mouse button click on the display entry. Highlighting is used to indicated which data values are currently selected. The **Options** menu also contains commands to select or deselect all displayed values. The selected values can be copied into the cut-and-paste (clipboard) buffer with the `CTRL-c` key combination, or the **Options|Copy** menu command.

The data value selection mechanism may also be used in data value formatting control. The **Options|Format** menu command brings up a **Format** dialog box to change the justification and format specification string. (The latter is the conversion string passed to the Tcl `format` command, which uses the C `printf` format codes.) If the **Adjust:Selected** radiobutton is active, then the specification will be applied to only the currently selected (highlighted) data values. Alternately, if **Adjust:All** is active, then the specification will be applied to all data values, and will additionally become the default specification.

A right mouse button click on a display entry will select that entry, and bring up the **Format** with the justification and format specifications of the selected entry. These specifications, with any revisions, may then be applied to all of the selected entries.

The menu selection **File|Exit** terminates the **mmDataTable** application. The menu **Help** provides the usual help facilities.

11 Data Graph Display: mmGraph



Overview

The application **mmGraph** provides a data display service similar to that of **mmDataTable** (Sec. 10). The usual data source is a running solver, but rather than the textual output provided by **mmDataTable**, **mmGraph** produces 2D line plots. **mmGraph** also stores the data it receives, so it can produce multiple views of the data and can save the data to disk. Postscript output is also supported.

Launching

mmGraph may be started either by selecting the **mmGraph** button on **mmLaunch** or from the command line via

```
tclsh oommf.tcl mmGraph [-console] [-nonet]
```

The `-nonet` option disables the socket server portion of **mmGraph**, in which case data may only be input from a file.

Inputs

Input to **mmGraph** may come from either a file in the ODT format (Sec. 17.2), or from a client application (typically a running solver). (The latter is disabled by the `-nonet` command line option.) The **File|Open...** dialog box is used to select an input file. Receipt of data from client applications is the same as for **mmDataTable** (Sec. 10). In either case, input data are appended to any previously held data.

Outputs

Unlike **mmDataTable**, **mmGraph** internally stores the data sent to it. This data may be written to disk via the **File|Save As...** dialog box. If the file specified already exists, then **mmGraph** output is appended to that file. The output is in the tabular ODT format described in Sec. 17.2. By default, all data currently held by **mmGraph** is written, but the **Save: Selected Data** option presented in the **File|Save As...** dialog box causes the output to be restricted to those curves currently selected for display. In either case, the graph display limits do *not* affect the output.

The save operation writes records that are held by **mmGraph** at the time the **File|Save As...** dialog box **OK** button is invoked. Additionally, the **Auto Save** option in this dialog box may be used to automatically append to the specified file each new data record as it is received by **mmGraph**. The appended fields will be those chosen at the time of the save operation, i.e., subsequent changing of the curves selected for display does not affect the automatic save operation. The automatic save operation continues until either a new output file is specified, the **Options|Stop autosave** control is invoked, or **mmGraph** is terminated.

The **File|Print...** dialog is used to produce a Postscript file of the current graph. On Unix systems, the output may be sent directly to a printer by filling the **Print to:** entry with the appropriate pipe command, e.g., `|lpr`. (The exact form is system dependent.)

Controls

Graphs are constructed by selecting any one item off the **X-Axis** menu, and any number of items off the **Y-Axis** menu. Either of these menus may be detached by selecting the dashed rule at the top of the list. Sample results are shown in the figure at the start of this section.

When **mmGraph** is first launched, the axis menus are both empty. They are dynamically built based on the data received by **mmGraph**. By default, the graph limits and labels are automatically set based on the data. The x-axis label is set using the selected item data label and measurement unit (if any). The y-axis label is the measurement unit of the first y-axis item selected. To change the default y-axis label, deselect all y-axis items and select a different first item.

The **Options|Configure...** dialog box allows the user to override the default settings. To set the axis labels, deselect the **Auto Label** option in this dialog box, and fill in the **X Label** and **Y Label** fields as desired. The axis limits can be set in a similar fashion. In addition, if an axis limit is left empty, a default value (based on all selected data) will be used.

As mentioned earlier, **mmGraph** stores all data it receives. Over the course of a long run, the amount of data stored can grow to many megabytes. This storage can be limited by specifying a positive value for the **Point buffer size** entry in the **Options|Configure...** dialog box. The oldest records are removed as necessary to keep the total number of records stored under the specified limit. A zero value for **Point buffer size** is interpreted as no limit. (The storage size of an individual record depends upon several factors, including the number of items in the record and the version of Tcl being used.) Data erasures may not be immediately reflected in the graph display.

At any time, the point buffer storage may be completely emptied with the **Options|clear Data** command. The **Options|Stop autosave** selection will turn off the auto save feature, if currently active. Also on this menu is **Options|Rescale**, which autoscales the graph axis limits from the selected data. This command ignores but does not reset the "Auto Scale" settings in the **Options|Configure...** dialog box.

The last command on the options menu is **Options|Key**, which toggles the key (legend) display on and off. The key may also be repositioned by dragging with the left mouse button.

Two other controls are also available only through the mouse. If the mouse pointer is positioned over a drawn item in the graph, holding down any mouse button will bring up the coordinates of that point. The coordinates displayed are the coordinates of a point on a drawn line, which are not necessarily the coordinates of a plotted data point. (The data points are plotted at the endpoints of each line segment.) The coordinate display is cleared when the mouse button is released.

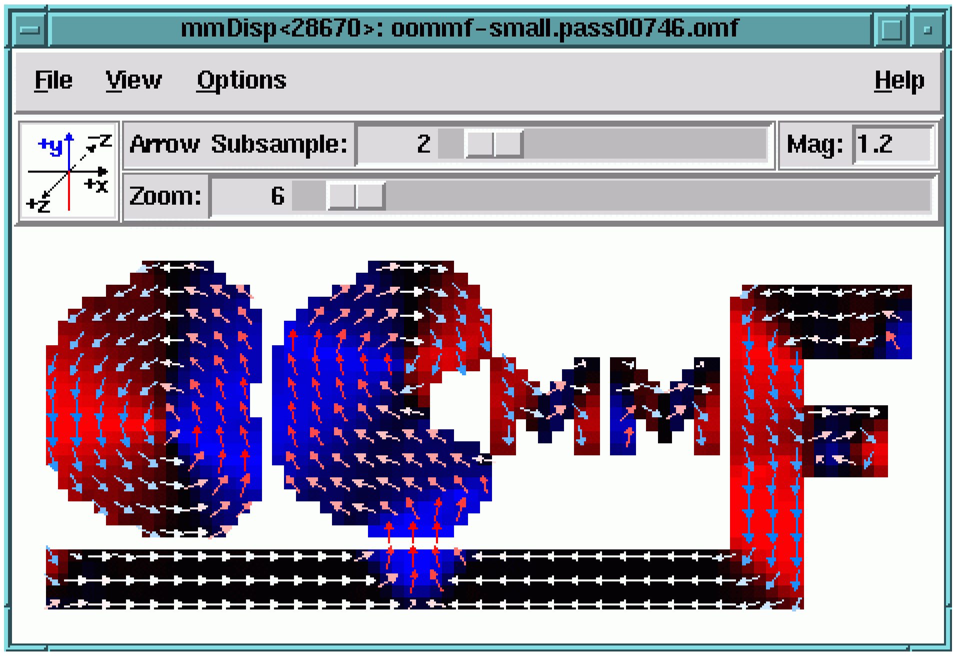
One vertical and one horizontal rule (line) are also available. Initially, these rules are tucked and hidden against the left and bottom graph axes, respectively. Either may be repositioned by dragging with the left mouse button.

The menu selection **File|Exit** terminates the **mmGraph** application. The menu **Help** provides the usual help facilities.

Details

The **X-Axis** and **Y-Axis** menus are configured based on incoming data. As a result, both menus are initially empty. If a graph widget is scheduled to receive data only upon equilibrium events in the solver, it may be a long time after starting a problem in the solver before the graph widget can be configured. Because **mmGraph** keeps all data up to the limit imposed by the *Point buffer size*, data loss is usually not a problem. Of more importance is the fact that automatic data saving can not be set up until the first data point is received. As a workaround, the solver initial state may be sent interactively as a dummy point to initialize the graph widget axes menus. (You may turn off the **Interactive** schedule connection after sending this data point.) Select the desired quantities off the axes menus, and use the **Options|clear Data** command to remove the dummy point from **mmGraph**'s memory. The **File|Save As...** dialog box may then be used—with the **Auto Save** option enabled—to write out an empty table with proper column header information. Subsequent data will be written to this file as it arrives.

12 Vector Field Display: mmDisp



Overview

The application **mmDisp** displays two-dimensional spatial distributions of three-dimensional vectors (i.e., vector fields). It can load vector fields from files in a variety of formats, or it can accept vector field data from a client application, typically a running solver. **mmDisp** offers a rich interface for controlling the display of vector field data, and can also save the data to a file and produce Postscript print output.

Launching

mmDisp may be started either by selecting the **mmDisp** button on **mmLaunch**, or from the command line via

```
tclsh oommf.tcl mmDisp [-server] [-version] \
     [-console] [--] [<filename>]
```

The command line option `-server` makes **mmDisp** a server supplying vector field display services to other applications. Without that option **mmDisp** only displays vector field data loaded from files. The `-server` option is automatically included when **mmDisp** is launched by **mmLaunch**.

If the option `-version` is present on the command line, version information is printed, and **mmDisp** exits immediately.

If a filename is supplied on the command line, **mmDisp** takes it to be the name of a file containing vector field data for display. That file will be opened on startup.

Inputs

Input to **mmDisp** may come from either a file or from a client application (typically a running solver), in any of the vector field formats described in Sec. 17.3. Other file formats can also be supported if a translation filter program is available.

Client applications that send data to **mmDisp** control the flow of data. The user, interacting with the **mmDisp** window, determines how the vector field data are displayed.

File input is initiated through the **File|Open...** dialog box. Several example files are included in the OOMMF release in the directory `app/mmdisp/examples`. When the **Browse** button is enabled, the "Open File" dialog box will remain open after loading a file, so that multiple files may be displayed in sequence. The **Auto** configuration box determines whether the vector subsampling or the zoom factor of the display should be determined automatically (based on the data in the file and the current display window size), or whether their values should be held constant while loading the file.

mmDisp permits local customization allowing for automatic translation from other file formats into one of the vector field formats (Sec. 17.3) that **mmDisp** recognizes. When loading a file, **mmDisp** compares the file name to a list of glob-style patterns. These patterns typically match on the filename extension. An example pattern is `*.gz`. The assumption is that the pattern identifies files containing data in a particular format. For each pattern in the list, there is a corresponding translation program. **mmDisp** calls on that program as a filter which takes data in one format from standard input and writes to standard output the same data in one of the formats supported by **mmDisp**. In its default configuration, **mmDisp** recognizes the pattern `*.gz` and invokes the translation program `gzip -dc` to perform the "translation." In this way, support for reading gzip compressed files is "built in" to **mmDisp** on any platform where the `gzip` program is installed.

New patterns and translation programs may be added to **mmDisp** by the usual method of local customization (Sec. 2.3.3). The command to add to the customization file is of the form

```
Oc_Option Add mmDisp Input filters {{*.gz {gzip -dc}}}
```

The final argument in this command is a list of pairs. The first element in each pair is the filename pattern. The second element in each pair is the command line for launching the corresponding translation program. If a program `foo` were known to translate a file format identified by the extension `.bar` into the OVF file format, that program could be made known to **mmDisp** by changing the above customization command to:

```
Oc_Option Add mmDisp Input filters {{*.gz {gzip -dc}} {*.bar foo}}
```

Outputs

The vector field displayed by **mmDisp** may be saved to disk via the **File|Save As...** dialog box. The output is in the OVF format (Sec. 17.3.1). The OVF file options may be set by selecting the appropriate radio buttons in the OVF File Options panel. The **Title** and **Desc** fields may be edited before saving. Enabling the **Browse** button allows for saving multiple files without closing the "Save File" dialog box.

The **File|Print...** dialog is used to produce a Postscript file of the current display. On Unix systems, the output may be sent directly to a printer by filling the **Print to:** entry with the appropriate pipe command, e.g., `|lpr`. (The exact format is system dependent.)

To produce bitmap output, save the file to disk in the OVF format, and use the `avf2ppm` (Sec. 15.1) utility to do the conversion.

Controls

The **View** menu provides high-level control over how the vector field is placed in the display window. The menu selection **View|Wrap Display** resizes the display window so that it just contains the entire vector field surrounded by a margin. **View|Fill Display** resizes the vector field until it fills the current size of the display window. If the aspect ratio of the display window does not match the aspect ratio of the vector field, a larger than requested margin appears along one edge to make up the difference. **View|Rotate ccw** and **View|Rotate cw** rotate the display one quarter turn counter-clockwise and clockwise respectively. The display window also rotates, so that the portion of the vector field seen and any margins are preserved (unless the display of the control bar forces the display window to be wider). **View|reDraw** allows the user to invoke a redrawing of the display window.

The vector field may be rescaled ("zoomed") and moved so that only a portion of it is visible in the display window. When a portion of the vector field lies outside the display window, scrollbars appear that may be used to translate the vector field so that different portions are visible in the display window. On systems that have a middle mouse button, clicking the middle button on a point in the display window translates the vector field so that the selected point is centered within the display window.

The mouse may also be used within the display window to rescale the display. A click and drag with the left mouse button displays a red rectangle that changes size as the mouse is dragged. When the left mouse button is released, the vector field is rescaled so that the portion of the display window within the red rectangle expands until it reaches the edges of the display window. Both dimensions are scaled by the same amount so there is no distortion of the vector field. Small red arrows on the sides of the red rectangle indicate which dimension will expand to meet the display window boundaries upon release of the left mouse button. After the rescaling, the red rectangle remains in the display window briefly, surrounding the same region of the vector field, but at the new scale.

A click and drag with the right mouse button displays a blue rectangle that changes size as the mouse is dragged. When the right mouse button is released, the vector field is rescaled so that all of the vector field currently visible in the display window fits in the size of the blue rectangle. Both dimensions are scaled by the same amount so there is no distortion of the vector field. Small blue arrows on the sides of the blue rectangle indicate the dimension in which the vector field will shrink to exactly transform the display window size to the blue rectangle size. After the rescaling, the blue rectangle remains in the display window briefly, surrounding the same region of the vector field, now centered in the display window, and at the new scale.

mmDisp remembers the previous display scale. To revert to the previous scale, the user may hit the `ESC` key. This is a limited "Undo" feature.

The menu selection **Options|Configure...** brings up a dialog box through which the user may control many features of the vector field display. Vectors in the vector field may be displayed as arrows, pixels, or both. The **Arrow** and **Pixel** buttons in the **Plot type** column on the left of the dialog box enable each type of display.

Columns 2–4 in the Configure dialog box control the use of color. Both arrows and pixels may be independently colored to indicate some quantity. The **Color Quantity** column controls which scalar quantity the color of the arrow or pixel represents. The x, y, or z components of the vector may be selected, or the in-plane xy-angle of the vector from the positive x-axis. On regularly gridded data the vector field divergence is also available for display.

The assignment of a color to a quantity value is determined by the **Colormap** selected. Colormaps are labeled by a sequence of colors that are mapped across the range of the selected quantity. For example, if the "Red-Black-Blue" colormap is applied to the **Color Quantity** "z", then vectors pointing into the xy-plane ($z < 0$) are colored red, those lying in the plane ($z = 0$) are colored black, and those pointing out of the plane ($z > 0$) are colored blue. Values between the extremes are colored with intermediate colors, selected using a discretization determined by the **# of Colors** value. This value limits the use of potentially limited color resources, and can be used to achieve some special coloring effects. (Note: The *xy*-angle quantity is bested viewed with a colormap that begins and ends with the same color, e.g., "Red-Green-Blue-Red.")

When there are many vectors in a vector field, a display of all of them can be more confusing than helpful. The **Subsample Rate** column allows the user to request that only a sampling of vectors from the vector field be displayed. The **Subsample Rate** value is roughly the number of vectors along one spatial dimension of the vector field which map to a single displayed vector (arrow or pixel). Each vector displayed is an actual vector in the vector field—the selection of vectors for display is a sampling process, not an averaging or interpolation process. The subsample rates for arrows and pixels may be set independently. A subsample rate of 0 is interpreted specially to display all data. (This is typically much quicker than subsampling at a small rate, e.g., 0.1.)

An arrow or pixel that represents a vector is by default displayed at a size that covers the portion of the vector field that that displayed vector is intended to represent, given the current subsample rate. Following this default, arrows do not significantly overlap each other, yet all portions of the vector field that are magnetically active have a representation in the display. The **Magnification** column allows the user to modify the size of the displayed arrows or pixels (independently). A **Magnification** of 1 is the default size. By changing to a larger or smaller **Magnification** value, the user may request arrows or pixels larger or smaller than the default size.

Lower in the **Option|Configure...** dialog box, the user may also select whether or not a bounding polygon is displayed, what margin (in pixels) should be maintained around the vector field, and what background color the display window should use.

None of the changes made by the user in the **Options|Configure...** dialog box affect the display window until either the **Apply** or **OK** button is selected. If the **OK** button is selected, the dialog box is also dismissed. The **Cancel** button dismisses the dialog without changing the display window.

The other item under the **Options** menu is a checkbutton that toggles the display of a control bar. The control bar offers alternative interfaces to some of the operations available from the **Options|Configure...** dialog box and the **View** menu. On the left end of the control bar is a display of the coordinate axes. These axes rotate along with the vector field in the display window to identify the coordinate system of the display, and are color coded to agree with the pixel (if active) or arrow coloring. A click of the left mouse button on the coordinate axes causes a counter-clockwise rotation. A click of the right mouse button on the coordinate axes causes a clockwise rotation.

To the right of the coordinate axes are two rows of controls. The top row allows the user to control the subsample rate and magnification of the arrow display. The subsample rate may be modified either by direct entry of a new rate, or by manipulation of the slider.

The lower row offers an alternative means to control the rescaling of the vector field in the display. The **Zoom** value roughly corresponds to the number of pixels per vector in the vector field. This value may be entered directly, or modified by manipulation of the slider. This alternative interface provides a more precise control over the rescaling of the vector field display than mouse operations provide.

Several keyboard shortcuts are available as alternatives to menu- or mouse-based operations. The effect of a key combination depends on which subwindow of **mmDisp** is active. The `TAB` key may be used to change the active subwindow. The `SHIFT-TAB` key combination also changes the active subwindow, in reverse order.

When the active subwindow is the display window, the following key combinations are active:

- `CTRL-o` – same as menu selection **File|Open...**
- `CTRL-s` – same as menu selection **File|Save as...**
- `CTRL-p` – same as menu selection **File|Print...**
- `CTRL-w` – same as menu selection **View|Wrap Display**
- `CTRL-f` – same as menu selection **View|Fill Display**
- `HOME` – First fill, then wrap the display.
- `CTRL-r` – same as menu selection **View|Rotate ccw**
- `SHIFT-CTRL-r` – same as menu selection **View|Rotate cw**
- `INSERT` – increase arrow subsample by 1
- `DEL` – decrease arrow subsample by 1
- `SHIFT-INSERT` – increase arrow subsample by factor of 2
- `SHIFT-DEL` – decrease arrow subsample by factor of 2
- `PAGEUP` – increase the zoom value by a factor of 1.149
- `PAGEDOWN` – decrease the zoom value by a factor of 1.149
- `SHIFT-PAGEUP` – increase the zoom value by factor of 2

- `SHIFT-PAGEDOWN` – decrease the zoom value by factor of 2
- `ESC` – revert to previous zoom value

When the active subwindow is the control bar's coordinate axes display, the following key combinations are active:

- `LEFT` – same as menu selection **View|Rotate ccw**
- `RIGHT` – same as menu selection **View|Rotate cw**

When the active subwindow is any of the control bar's value entry windows – arrow subsample, magnification, or zoom, the following key combinations are active:

- `ESC` – undo uncommitted value (displayed in red)
- `RETURN` – commit entered value

When the active subwindow is either of the control bar's sliders—arrow subsample or zoom—the following key combinations are active:

- `LEFT` – slide left (decrease value)
- `RIGHT` – slide right (increase value)
- `ESC` – undo uncommitted value (displayed in red)
- `RETURN` – commit current value

Of course the usual keyboard access to the menu items is also available.

The menu selection **File|Clear** clears the display window. The menu selection **File|Exit** terminates the **mmDisp** application. The menu **Help** provides the usual help facilities.

Details

The selection of vectors for display according to the **Subsample Rate** differs depending on whether or not the data lies on a regular grid. If so, the **Subsample Rate** takes integer values and determines the ratio of data points to displayed points. For example, a value of 5 means that every fifth vector on the grid is displayed. This means that the number of vectors displayed is 25 times fewer than the number of vectors on the grid.

For an irregular grid of vectors, an average cell size is computed, and the **Subsample Rate** takes values in units of 0.1 times the average cell size. A square grid of that size is overlaid on the irregular grid. For each cell in the square grid, the data vector from the irregular grid closest to the center of the square grid cell is selected for display. The vector is displayed at its true location in the irregular grid, not at the center of the square grid cell. As the subsample rate changes, the set of displayed vectors also changes, which can in some circumstances substantially change the appearance of the displayed vector field.

Using mmDisp as a WWW browser helper application

You may configure your web browser to automatically launch **mmDisp** when downloading an OVF file. The exact means to do this depends on your browser, but a couple of examples are presented below.

In Netscape Navigator 4.X, bring up the **Edit|Preferences...** dialog box, and select the `Category` **Navigator|Applications** subwindow. Create a **New Type**, with the following fields:

Description of type: OOMMF Vector Field

MIME Type: application/x-oommf-vf

Suffixes: ovf omf ohf obf svf

Application: *wish oommfroot*/oommf.tcl -fg mmDisp "*arg*"

On Windows platforms, the `Suffixes` field is labeled `File Extension`, and only one file extension may be entered. Files downloaded from a web server are handled according to their MIME Type, rather than their file extension, so that restriction isn't important when web browsing. If you wish to have files on the local disk with all the above file extensions recognized as OOMMF Vector Field files, you must repeat the **New Type** entry for each file extension. In the `Application` field, the values of *wish*, *oommfroot*, and *arg* vary with your platform configuration. The value of *wish* is the full path to the **wish** application on your platform (see Section 5). On Unix systems, *wish* may be omitted, assuming that the `oommf.tcl` script is executable. If *wish* is not omitted on Unix systems, Netscape will issue a security warning each time it opens an OOMMF Vector Field file. The value of *oommfroot* should be the full path to the root directory of your OOMMF installation. The value of *arg* should be "%1" on Windows and "%s" on Unix. The MIME type "application/x-oommf-vf" must be configured on any HTTP server which provides OOMMF Vector Field files as well.

For Microsoft Internet Explorer 3.X, bring up the **View|Options...** dialog box, and select the **Program** tab. Hit the **File Types...** button, followed by the **New Type...** button. Fill the resulting dialog box with

Description of type: OOMMF Vector Field

Associated extension: ovf

Content type (MIME): application/x-oommf-vf

You may also disable the **Confirm open after download** checkbutton if you want. Then hit the **New...** button below the `Actions:` window, and in the pop-up fill in

Action: open

Application used to perform action:

wish oommfroot/oommf.tcl -fg mmDisp "%1"

Hit **OK**, **Close**, **Close** and **OK**. Replace *wish* and *oommfroot* with the appropriate paths on your system (cf. Section 5). This will set up an association on files with the .ovf extension. Internet Explorer 3.X apparently ignores the HTML Content Type field, so you must repeat this process for each file extension (.ovf, .omf, .ohf, .obf and .svf) that you want to recognize. This means, however, that Internet Explorer will make the appropriate association even if the HTML server does not properly set the HTML Content Type field.

Microsoft Internet Explorer 4.X is integrated with the Windows operating system. Internet Explorer 4.X doesn't offer any means to set up associations between particular file types and the applications which should be used to open them. Instead, this association is configured within the Windows operating system. To set up associations for the OOMMF Vector Field file type on Windows 95 or Windows NT 4.0, select **Settings|Control Panel** from the **Start** menu. The Control Panel window appears. Select **View|Options...** to display a dialog box. A Windows 98 shortcut to the same dialog box is to select **Settings|Folder Options...** from the **Start** menu. Select the **File Types** tab and proceed as described above for Internet Explorer 3.X. Depending on the exact release/service patch of your Windows operating system, the exact instructions may vary.

With the 1.x releases of OOMMF, you may use the alternate application line

oommfroot/app/mmdisp/*platformname*/mmdisp.exe "*arg*"

where *oommfroot* and *arg* are as in the discussion above and *platformname* is the name which OOMMF assigns to your platform type, as reported by the script `platform.tcl`. Also, the file extension `.exe` is for Windows platforms only. This alternative specification of the application for opening OOMMF Vector Field files should launch a little bit faster than the more generic command line given earlier.

Once you have your browser configured, you can test it on the muMag 1st Standard Problem report page,

http://www.ctcms.nist.gov/%7Erdm/std1/vectorcompare.html.

13 Data Archive: mmArchive

Status: Opened oommf-small.odt

Overview

The application **mmArchive** provides automated vector field and data table storage services. Although **mmDisp** (Sec. 12) and **mmGraph** (Sec. 11) are able to save such data under the direction of the user, there are situations where it is more convenient to write data to disk without interactive control.

mmArchive does not present a user interface window of its own, but like **mmSolve2D** (Sec. 9) relies on **mmLaunch** (Sec. 6) to provide an interface on its behalf. Because **mmArchive** does not require a window, it is possible on Unix systems to bring down the X (window) server and still keep **mmArchive** running in the background.

Launching

mmArchive may be started either by selecting the **mmArchive** button on **mmLaunch**, or from the command line via

```
tclsh oommf.tcl mmArchive [-tk <0|1>] [-console]
```

When the **mmArchive** button of **mmLaunch** is invoked, **mmArchive** is launched with the `-tk 0` option. This allows **mmArchive** to continue running if the X window server is killed. The `-tk 1` option is useful only for enabling the `-console` option for debugging.

As noted above, **mmArchive** depends upon **mmLaunch** to provide an interface. The entry for an instance of **mmArchive** in the **Threads** column of any running copy of **mmLaunch** has a checkbutton next to it. This button toggles the presence of a user interface window through which the user may control that instance of **mmArchive**.

Inputs

mmArchive accepts vector field and data table style input from client applications (typically running solvers) on its network (socket) interface.

Outputs

The client applications that send data to **mmArchive** control the flow of data. **mmArchive** copies the data it receives into files specified by the client. There is no interactive control to select the names of these output files. A simple status line shows the most recent vector file save, or data table file open/close event.

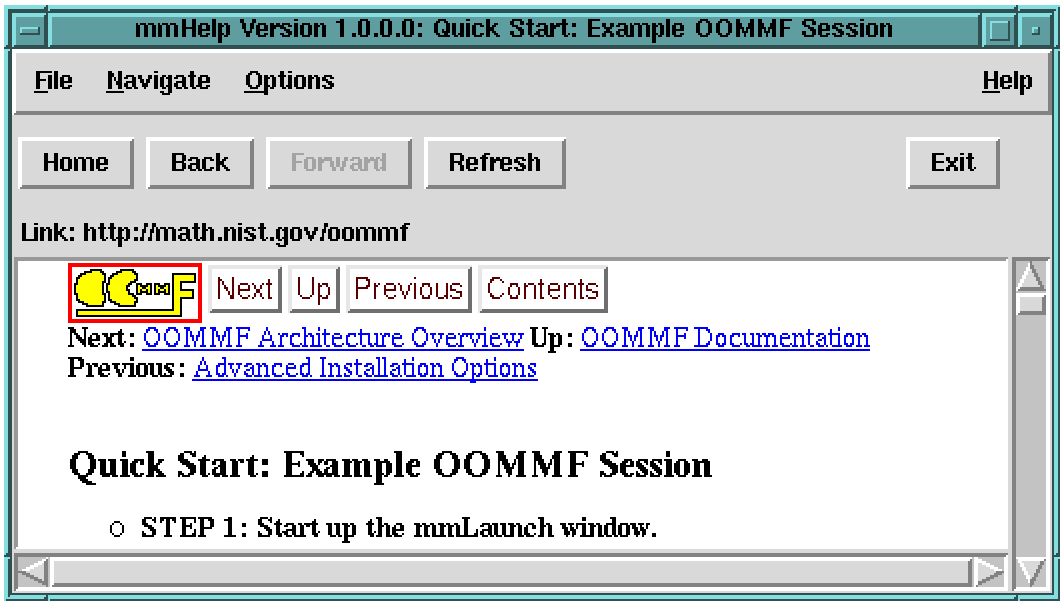
Controls

The **Exit** in the **mmLaunch**-supplied user interface terminates **mmArchive**. Simply closing the user interface window does not terminate **mmArchive**, but only hides the control window. To kill **mmArchive** the **Exit** button must be pressed.

Known Bugs

mmArchive appends data table output to the file specified by the source client application (e.g., a running solver). If, at the same time, more than one source specifies the same file, or if the the same source sends data table output to more than one instance of **mmArchive**, then concurrent writes to the same file may corrupt the data in that file. It is the responsibility of the user to ensure this does not happen; there is at present no file locking mechanism in OOMMF to protect against this situation.

14 Documentation Viewer: mmHelp



Link: http://math.nist.gov/oommf

Next: OOMMF Architecture Overview Up: OOMMF Documentation
Previous: Advanced Installation Options

Quick Start: Example OOMMF Session

- **STEP 1: Start up the mmLaunch window.**

Overview

The application **mmHelp** manages the display and navigation of hypertext (HTML) help files. It presents an interface similar to that of World Wide Web browsers.

Although **mmHelp** is patterned after World Wide Web browsers, it does not have all of their capabilities. **mmHelp** displays only a simplified form of hypertext required to display the OOMMF help pages. It is not able to display many of the advanced hypertext features provided by modern World Wide Web browsers. In the current release, **mmHelp** is not able to follow `http:` URLs. It only follows `file:` URLs.

OOMMF software can be customized (See Sec. 2.3.3) to use another program to display the HTML help files.

Launching

mmHelp may be launched from the command line via

```
tclsh oommf.tcl mmHelp [<URL>] [-console]
```

The command line argument `<URL>` is the URL of the first page (or home page) to be displayed. If no URL is specified, **mmHelp** displays the Table of Contents of the *OOMMF User's Guide* by default.

Controls

Each page of hypertext is displayed in the main **mmHelp** window. Words which are underlined and colored blue are hyperlinks which **mmHelp** knows how to follow. Words which are underlined and colored red are hyperlinks which **mmHelp** does not know how to follow. Moving the mouse over a hyperlink displays the target URL of the hyperlink in the **Link:** line above the display window. Clicking on a blue hyperlink will follow the hyperlink and display a new page of hypertext.

mmHelp keeps a list of the viewed pages in order of view. Using the **Back** and **Forward** buttons, the user may move backward and forward through this list of pages. The **Home** button causes the first page to be displayed, allowing the user to start again from the beginning. These three buttons have corresponding entries in the **Navigate** menu.

The menu selection **File|Refresh**, or the **Refresh** button causes **mmHelp** to reload and redisplay the current page. This may be useful if the display becomes corrupted, or for repeatedly loading a hypertext file which is being edited.

When **mmHelp** encounters hypertext elements it does not recognize, it will attempt to work around the problem. However, in some cases it will not be able to make sense of the hypertext, and will display an error message. Documentation authors should take care to use only the hypertext elements supported by **mmHelp** in their documentation files. Users should never see such an error message.

mmHelp displays error messages in one of two ways: within the display window, or in a separate window. Errors reported in the display window replace the display of the page of hypertext. They usually indicate that the hypertext page could not be retrieved, or that its contents are not hypertext. File permission errors are also reported in this way.

Errors reported in a separate window are usually due to a formatting error within the page of hypertext. Selecting the **Continue** button of the error window instructs **mmHelp** to attempt to resume display of the hypertext page beyond the error. Selecting **Abort** abandons further display.

The menu selection **Options|Font scale...** brings up a dialog box through which the user may select the scale of the fonts to use in the display window, relative to their initial size.

The menu selection **File|Exit** or the **Exit** button terminates the **mmHelp** application. The menu **Help** provides the usual help facilities.

15 Command Line Utilities

This section documents a few utilities distributed with OOMMF that are run from the command line (Unix shell or Windows DOSprompt), which are typically used in pre- or post-processing of data associated with a micromagnetic simulation.

15.1 Making Bitmaps from Vector Fields: avf2ppm

The **avf2ppm** utility converts a collection of vector field files (e.g., `.omf`, `.ohf`) into color bitmaps suitable for inclusion into documents or collating into movies. The command line arguments control filename selection, while plain-text configuration files, modeled after the **mmDisp** (Sec. 12) configuration dialog box, specify conversion parameters.

Command syntax

The **avf2ppm** launch command is:

```
tclsh oommf.tcl -fg avf2ppm [-h] [-tk 0] [-v <level>] [-f] \
   [-type <P3|P6>] [-config <filename>] [-ipat <glob match spec>] \
   [-opatexp <rexp match spec>] [-opatsub <sub spec>] \
   [-filter <program(s)>] [--] [<infile> ...]
```

where

-h Display help message.

-tk 0 If `-tk 0` is used, then **avf2ppm** will not use or initialize Tk. Tk is only needed to convert background color requests (see `misc,background` in the configuration file discussion below) from symbolic form to hexadecimal representation (`#RRGGBB`). If the background color is specified using the hexadecimal format, then Tk is not needed, and **avf2ppm** may be run with `-tk 0`.

-v <level> Verbosity (informational message) level, with 0 generating only error messages, and larger numbers generating additional information. The `<level>` value is an integer, defaulting to 1.

-f Force overwriting of existing (output) files. By default, if **avf2ppm** tries to create a file, say `foo.ppm`, that already exists, it generates instead a new name of the form `foo.ppm-000`, or `foo.ppm-001`, . . . , or `foo.ppm-999`, that doesn't exist and writes to that instead. The `-f` flag disallows alternate filename generation, and overwrites `foo.ppm` instead.

-type <P3|P6> Specify the output image file format. Currently support formats are the true color *Portable Pixmap* (PPM) formats P3 (ASCII) and P6 (binary). The default is P6.

-config <filename> Specifies a user (per-run) configuration file to use to specify the image conversion parameters. The configuration file is discussed in detail below.

-ipat <glob match spec> Specify input files using ("glob-style") wildcards. Mostly useful in DOS.

-opatexp <rexp match spec> Specify the "regular expression" applied to input filenames to determine portion to be replaced in generation of output filenames. Default: `(\.[^.]?[^.]?[^.]?$|$)`

-opatsub <sub spec> The string with which to replace the portion of input filenames matched by the `-opatsub` during output filename generation. Default: `.ppm`

-filter <program(s)> Post-processing application(s) to run on each **app2ppm** output file.

-- (Two dashes.) Marks the end of options. Anything after this will be interpreted as input vector field files to process.

<infile> ... List of input files to process.

The file specification options require some explanation. Input files may be specified either by an explicit list (`[<infile> ...]`), or by giving a wildcard pattern, e.g., `-ipat *.omf`, which is expanded in the usual way by **avf2ppm** (using the Tcl command `glob`). Unix shells (sh, csh, etc.) automatically expand wildcards before handing control over to the invoked application, so the `-ipat` option is not needed (although it is useful in case of a "command-line too long" error). DOS does not do this expansion, so you must use `-ipat` to get wildcard expansion in Windows.

As each input file is processed, a name for the output file is produced from the input filename by rules determined by handing the `-opatexp` and `-opatsub` expressions to the Tcl `regsub` command. Refer to the Tcl `regsub` documentation for details, but essentially whatever portion of the input filename is matched by the `-opatexp` expression is removed and replaced by the `-opatsub` string. The default `-opatexp` expression matches against any filename extension of up to 3 characters, and the default `-opatsub` string replaces this with the extension `.ppm`.

If you have command line image processing "filter" programs, e.g., **ppmtogif** (part of the NetPBM package), then you can use the `-filter` option to pipe the output of **avf2ppm** through that filter before it is written to the output file specified by the `-opat` expressions. If the processing changes the format of the file, (e.g., **ppmtogif** converts from PPM to GIF), then you will likely want to specify a `-opatsub` different from the default.

Here is an example that processes all input files with the `.omf` extension, sending the output through **ppmtogif** before saving the results in a files with the extension `.gif`:

```
tclsh oommf.tcl -fg avf2ppm -ipat *.omf -opatsub .gif \
   -filter ppmtogif
```

(On Unix, either drop the `-ipat` flag, or use quotes to protect the input file specification string from expansion by the shell, as in `'*.omf'`.) You may also pipe together multiple filters, e.g., `-filter 'ppmquant 256 | ppmtogif'`.

Configuration files

The details of the conversion process are specified by plain-text configuration files, with fields analogous to the entries in the **mmDisp** (Sec. 12) configuration dialog box. Each of the parameters is an element in an array named `plot_config`. The default values for this array are taken from the default configuration file `oommf/app/mmdisp/avf2ppm.def`, which is a Tcl script read during **avf2ppm** initialization.

The sample default configuration script shown in Fig. 1 can be used as a starting point for user (per-run) configuration files. Refer to this sample file and the mmDisp documentation (Sec. 12) as we discuss each element of the array `plot_config`. (See the Tcl documentation for details of the `array set` command.)

colormaps A list of valid colormaps known to the program. This entry is *not* user-configurable, and should not appear in user configuration files.

arrow,status Set to 1 to display arrows, 0 to not draw arrows.

arrow,colormap Select the colormap to use when drawing arrows. Should be one of the strings specified in the `colormaps` array element.

arrow,colorcount Number of discretization levels to use from the colormap. A value of zero will color all arrows with the first color in the colormap.

arrow,quantity Scalar quantity the arrow color is to represent. Supported values include `x`, `y`, and `z`. The **mmDisp** configuration dialog box will present the complete list of allowed quantities (which may be image dependent).

arrow,autosample If 1, then ignore the value of `arrow,subsample` and automatically determine a “reasonable” subsampling rate for the arrows. Set to 0 to turn off this feature.

arrow,subsample If `arrow,autosample` is 0, then subsample the input vectors at this rate when drawing arrows. A value of 0 for `arrow,subsample` is interpreted specially to display all data.

arrow,mag Size of the arrows relative to the default size (represented as 1.0).

arrow,antialias If 1, then each pixel along the edge of an arrow is drawn not with the color of the arrow, but with a mixture of the arrow color and the background color. This makes arrow boundaries appear less jagged, but increases computation time. Also, the colors used in the anti-aliased pixels are not drawn from the arrow or pixel colormap discretizations, so color allocation in the output bitmap may increase dramatically.

pixel,... Each pixel configuration element has interpretation analogous to the corresponding array configuration element, except that there is no `pixel,antialias` element, and the auto subsampling rate for pixels is considerably denser than for arrows.

misc,background Specify the background color, using any of the forms recognized by the Tk routine `Tk_GetColor`. In particular, you may use symbolic names such as white, black, green, or hexadecimal representations with the format `#RRGGBB` (for example, `#ffff00` is yellow).

misc,drawboundary If 1, then draw the bounding polygon, if any, as specified in the input vector field format file.

misc,margin The size of the border margin, in pixels.

misc,width, misc,height Maximum width and height of the output bitmap, in pixels. If `misc,crop` is enabled, then one or both of these dimensions may be shortened.

misc,crop If disabled (0), then any leftover space in the bitmap (of dimensions `misc,width` by `misc,height`) after packing the image are filled with the background color. If enabled (1), then the bitmap is cropped to just include the image (with the margin specified by `misc,margin`). **NOTE:** Some movie formats require that bitmap dimensions be multiples of 8 or 16. For such purposes, you should disable `misc,crop` and specify appropriate dimensions directly with `misc,width` and `misc,height`.

misc,zoom Scaling factor for the display. This is the same value as shown in the "zoom" box in the **mmDisp** control bar, and corresponds roughly to the number of pixels per vector in the (original, fully-sampled) vector field. If set to zero, then **avf2ppm** will automatically set the scaling so the image (with margins) just fits inside the area specified by `misc,width` and `misc,height`.

misc,rotation Rotation in degrees; either 0, 90, 180 or 360.

User (per-run) configuration files are specified on the command line with the `-config` option. To create a user configuration file, make a copy of the default `avf2ppm.def` configuration file, and edit it as desired in a plain text editor. You may omit any entries that you do not want to change from the default. (Each entry consists of a name + value pair, e.g., `misc,width 640`.) You may "layer" configuration files by specifying multiple user configuration files on the command line. These are processed from left to right, with the last value set for each entry taking precedence.

15.2 Bitmap File Format Conversion: any2ppm

The **any2ppm** program converts bitmap files of various formats into the Portable Pixmap (PPM) P3 (text) format. Supported input formats are PPM, BMP, and GIF. (Note: OOMMF support for BMP requires Tcl/Tk 8.0 or later.)

Command syntax

The **any2ppm** launch command is:

```
tclsh oommf.tcl -fg any2ppm [-h] [-noinfo] [-o <outfile>] \
   [<infile> ...]
```

where

-h Display help message.

-noinfo Suppress writing of progress info to stderr.

-o <outfile> Write output to <`outname`>; use "-" to pipe output to stdout. The default is to create a new file by stripping the extension, if any, off of each input filename, and appending `.ppm`. If the generated filename already exists, a "-000" or "-001" … suffix is appended.

<infile> ... List of input files to process.

```
array set plot_config {
    colormaps  { Red-Black-Blue Blue-White-Red Teal-White-Red \
            Black-Gray-White White-Green-Black Red-Green-Blue-Red }
    arrow,status       1
    arrow,colormap     Black-Gray-White
    arrow,colorcount   0
    arrow,quantity     z
    arrow,autosample   1
    arrow,subsample    10
    arrow,mag          1
    arrow,antialias    1
    pixel,status       1
    pixel,colormap     Teal-White-Red
    pixel,colorcount   225
    pixel,quantity     x
    pixel,autosample   1
    pixel,subsample    2
    pixel,mag          1
    misc,background    white
    misc,drawboundary  1
    misc,margin        10
    misc,width         640
    misc,height        480
    misc,crop          1
    misc,zoom          0
    misc,rotation      0
}
```

Figure 1: Sample default configuration script `avf2ppm.def`.

Tk Requirement: any2ppm uses the Tk `image` command in its processing. This requires that Tk be properly initialized, which in particular means that a valid display must be available. This is not a problem on Windows, where a desktop is always present, but on Unix this means that an X server must be running. The *Xvfb* virtual framebuffer can be used if desired. (Xvfb is an X server distributed with X11R6 that requires no display hardware or physical input devices.)

15.3 Vector Field File Format Conversion: avf2ovf

The **avf2ovf** program converts vector field files from any of the recognized formats (OVF, SVF, VIO; see Sec. 17.3) into the OVF format.

Command syntax

The **avf2ovf** launch command is:

```
tclsh oommf.tcl -fg avf2ovf [-h] [-ascii|-b4|-b8] [-reg|-irreg] \
   infile >outfile
```

where

-h Display help message.

-ascii|-b4|-b8 Specify output data format. The default is ASCII; -b4 selects 4-byte binary, -b8 selects 8-byte binary. (The OVF format has an ASCII header in all cases.)

-reg|-irreg Specify output grid structure. The default is -reg, which will output a regular (rectangular) grid if then input is recognized as a regular grid. The -irreg option forces irregular mesh style output.

infile Name of input file to process. Must be one of the recognized formats, OVF, SVF, or VIO.

>**outfile** Avf2ovf writes its output to stdout. Use the redirection operator ">" to send the output to a file.

The -ascii and -irreg options are useful for preparing files for import into other applications, because all non-data lines are readily identified by a leading "`#`," and each data line is a 6-tuple consisting of the node location and vector value. Pay attention, however, to the scaling of the vector value as specified by "`# valueunit`" and "`# valuemultiplier`" header lines.

For output format details, see the OVF file description (Sec. 17.3.1).

16 OOMMF Batch System

The OOMMF Batch System (OBS) provides a scriptable interface (`batchsolve.tcl`) to the same micromagnetic solver engine used by **mmSolve2D** (Sec. 9), and several Tcl scripts (`batchmaster.tcl`, `batchslave.tcl`, and `batchtask.tcl`) that provide support for complex job scheduling. All OBS script files are in the OOMMF distribution directory `app/mmsolve/scripts`.

Unlike most of the OOMMF package, the OBS is meant to be driven primarily from the command line or shell (batch) script. OBS applications are launched from the command line using the bootstrap application (Sec. 5).

16.1 Batch solver

Overview

The batch solver script `batchsolve.tcl` provides a simple command line interface to the OOMMF micromagnetic solver engine.

Launching

The batch solver is launched by the command line:

```
tclsh oommf.tcl batchsolve [-tk 0] [[-file] <filename.mif>] \
   [-no_interface] [-start_paused] [-end_paused] [-end_exit <0|1>] \
   [-restart <0|1>]
```

where

-tk 0 Do not load Tk or display a (blank) window,

[-file] <filename.mif> Immediately load and run the specified MIF file. The `[-file]` tag may be omitted if the first character of `filename` is not a hyphen.

-no_interface Do not register with the account service directory application. As a result, `batchsolve.tcl` will not show up on the threads list in **mmLaunch** (Sec. 6), and no interactive interface will be provided by **mmLaunch**. (However, if `batchsolve.tcl` is invoked from inside another script (e.g., `batchslave.tcl`), the calling script could provide an interface to the batch-controlled solver.)

-start_paused Pause solver after loading problem.

-end_paused Pause solver and enter event loop at bottom of `batchsolve.tcl` rather than just falling off the end (the effect of which will depend on whether or not Tk is loaded).

-end_exit <**0|1**> If 1, then explicitly call exit at bottom of `batchsolve.tcl`. The default is 1 if and only if a MIF file has been specified on the command line.

-restart <**0|1**> Determines solver behavior when a new problem is loaded. If 1, then the solver will look for ***`basename`***`.log` and ***`basename`***`*.omf` files to restart a previous run from the last saved state (where ***`basename`*** is the "Base Output Filename" specified in the input problem specification). If these files cannot be found, then a warning is issued and the solver falls back to the default behavior (equivalent to `-restart 0`) of starting the problem from scratch. The specified `-restart` setting holds for **all** problems fed to the solver, not just the first.

The input `filename.mif` file should contain a Micromagnetic Input Format (Sec. 17.1) (MIF) problem description, such as produced by **mmProbEd** (Sec. 7). The batch solver searches several directories for this file, including the current working directory, the `data` and `scripts` subdirectories, and parallel directories relative to the directories `app/mmsolve` and `app/mmpe` in the OOMMF distribution. Refer to the `mif_path` variable in `batchsolve.tcl` for the complete list.

If `batchsolve.tcl` registers itself with the account service directory application, it will appear in the **Threads** list of any running copy of **mmLaunch**, and **mmLaunch** will provide an interactive interface on behalf of `batchsolve.tcl`. Using this interface, `batchsolve.tcl` may be controlled in a manner similar to **mmSolve2D** (Sec. 9). The interface allows you to pause, un-pause, and terminate the current simulation, as well as to attach data display applications to monitor the solver's progress. If more interactive control is needed, **mmSolve2D** should be used.

Use the `-start_paused` switch to monitor the progress of `batchsolve.tcl` from the very start of a simulation. With this switch the solver will be paused immediately after loading the specified MIF file, so you can bring up the interactive interface and connect display applications before the simulation begins. Start the simulation by selecting the **Run** command from the interactive interface.

The `-end_paused` switch insures that the solver does not automatically terminate after completing the specified simulation. This is not generally useful, but may find application when `batchsolve.tcl` is called from inside a Tcl-only wrapper script.

Output

The output may be changed by a Tcl wrapper script (see Sec. 16.1), but the default output behavior of `batchsolve.tcl` is to write tabular text data and the magnetization state at equilibrium for each applied field step. The tabular data are appended to the file *`basename`*`.odt`, where *basename* is the "Base Output Filename" specified in the input MIF file. See the routine `GetTextData` in `batchsolve.tcl` for details, but at present the output consists of the solver iteration count, nominal applied field **B**, reduced magnetization **m**, and total energy. This output is in the ODT file format.

The magnetization data are written to a series of OVF (OOMMF Vector Field) files, *`basename`*`.field`*`nnnn`*`.omf`, where *`nnnn`* starts at `0000` and is incremented at each applied field step. (The ASCII header inside each file records the nominal applied field at that step.) These files are viewable using **mmDisp** (Sec. 12).

In addition, the solver automatically appends the input problem specification and miscellaneous runtime information to the log file *`basename`*`.log`.

Programmer's interface

In addition to directly launching `batchsolve.tcl` from the command line, it may also be sourced from inside another Tcl script, e.g., `batchslave.tcl`, that provides additional control structures. There are several variables and routines inside `batchsolve.tcl` that may be accessed and redefined from such a wrapper script to provide enhanced functionality.

Global variables

mif A Tcl handle to a global `mms_mif` object holding the problem description defined by the input MIF file.

solver A Tcl handle to the `mms_solver` object.

search_path Directory search path used by the `FindFile` proc (see below).

Refer to the source code and sample scripts for details on manipulation of these variables.

Batchsolve procs

The following Tcl procedures are designed for external use and/or redefinition:

SolverTaskInit Called at the start of each task.

BatchTaskRelaxCallback Called at each equilibrium state reached in the simulation.

SolverTaskCleanup Called at the conclusion of each task.

FindFile Searches the directories specified by the global variable `search_path` for a specified file. The default `SolverTaskInit` proc uses this routine to locate the requested input MIF file.

The first and third of these accept an arbitrary argument list (`args`), which is copied over from the `args` argument to `batchsolve.tcl` procs `BatchTaskRun` and `BatchTaskLaunch`. Typically one copies the default procs (as needed) into a task script, and makes appropriate modifications. You may (re-) define these procs either before or after sourcing `batchsolve.tcl`. See Sec. 16.2.3 for example scripts.

16.2 Batch scheduler

Overview

The OBS provides support for complex scheduling of multiple batch jobs via the two files `batchmaster.tcl` and `batchslave.tcl` plus a user defined task script. The task script may be modeled after the included `simpletask.tcl` or `multitask.tcl` sample scripts.

The OBS has been designed to control multiple sequential and concurrent micromagnetic simulations, but the scheduling scripts `batchmaster.tcl` and `batchslave.tcl` are completely general and may be used to schedule other types of jobs as well.

When `batchmaster.tcl` is run, it sources a task script that should modify the global object `$TaskInfo` to inform the master what tasks to perform and optionally how to launch slaves to perform those tasks. This is detailed in Sec. 16.2.2. After this, `batchmaster.tcl` launches all the specified slaves, initializes each with a slave initialization script, and then feeds tasks sequentially from the task list to the slaves. When a slave completes a task it reports back to the master and is given the next unclaimed task. If there are no more tasks, the slave is shut down. When all the tasks are complete, the master prints a summary of the tasks and exits. Note: At present, launching and controlling jobs off the local machine requires a working `rsh` command or equivalent.

Launching

The batch scheduler is launched by the command line:

```
tclsh oommf.tcl batchmaster [-tk 0] <task_script.tcl> \
      [server_host [server_port]]
```

where

-tk 0 Do not load Tk or display a (blank) window,

<task_script.tcl> is the user defined task (job) definition file,

server_host specifies the network address for the master to use (default is *localhost*),

server_port is the port address for the master (default is *0*, which selects an arbitrary open port).

The required <`task_script.tcl`> should be based on the example scripts in Sec. 16.2.3. If slaves are to be run on remote machines, then `server_host` **must** be set to the local machine's network name, and the `$TaskInfo` methods `AppendSlave` and `ModifyHostList` will need to be called from inside the task script. See Sec. 16.2.2 for details.

16.2.1 Batch master and slave specifics

The communication protocol between `batchmaster.tcl` and `batchslave.tcl` is evolving and is not described here. Check the source code for the latest details.

The command line to start the master script is shown above. The slave, which is launched by the master using instructions specified in the task script, takes a command line of the form

```
 tclsh oommf.tcl batchslave $server(host) $server(port) $slaveid \
       $passwd [aux_script [aux_args]]
```

The arguments `$server(host)` through `$passwd` are provided by the master and should be specified in the task script using the `%connect_info` percent substitution token. The `aux_script` is a script for the slave to source before processing any commands from the master. Typically this will be the micromagnetic batch mode script, `batchsolve.tcl`. The `aux_args` are additional arguments that will be passed to `aux_script`.

16.2.2 Batch task scripts

The master script creates an instance of a `BatchTaskObj` object with the name `$TaskInfo`. The task script uses method calls to this object to set up tasks to be performed. The only required call is to the `AppendTask` method, e.g.,

```
$TaskInfo AppendTask A "BatchTaskRun taskA.mif"
```

This method expects two arguments, a label for the task (here “A”) and a script to accomplish the task. The script will be passed across a network socket from the master to the slave, and then the script will be interpreted by the slave. (In particular, keep in mind that the file system seen by the script will be that of the machine on which the slave process is running.)

This example uses the default `batchsolve.tcl` procs to run the simulation defined by the `taskA.mif` MIF file. If you want to make changes to the MIF problem specifications on the fly, you will need to modify the default procs. This is done by creating a slave initialization script, via the call

```
$TaskInfo SetSlaveInitScript { <insert script here> }
```

The slave initialization script does global initializations, and also generally redefines the `SolverTaskInit` proc; optionally the `BatchTaskRelaxCallback` and `SolverTaskCleanup` procs may be redefined as well. At the start of each task `SolverTaskInit` is called by `BatchTaskRun` (in `batchsolve.tcl`), at each equilibrium `BatchTaskRelaxCallback` is executed, and at the end of each task `SolverTaskCleanup` is called. The first and third are passed the arguments that were passed to `BatchTaskRun`. A simple `SolverTaskInit` proc could be

```
 proc SolverTaskInit { args } {
    global mif basename outtextfile
    set A [lindex $args 0]
    set outbasename "$basename-A$A"
    $mif SetA $A
    $mif SetOutBaseName $outbasename
    set outtextfile [open "$outbasename.odt" "a+"]
    puts $outtextfile [GetTextData header \
          "Run on $basename.mif, with A=[$mif GetA]"]
 }
```

This proc receives the exchange constant `A` for this task on the argument list, and makes use of the global variables `mif` and `basename`. (Both should be initialized in the slave initialization script outside the `SolverTaskInit` proc.) It then stores the requested value of `A` in the `mif` object, sets up the base filename to use for output, and opens a text file to which tabular data will be appended. The handle to this text file is stored in the global `outtextfile`, which is closed by the default `SolverTaskCleanup` proc. A corresponding task script could be

```
$TaskInfo AppendTask "A=13e-12 J/m" "BatchTaskRun 13e-12"
```

which runs a simulation with `A` set to 13×10^{-12} J/m. This example is taken from the `multitask.tcl` script in Sec. 16.2.3.

If you want to run more than one task at a time, then the `$TaskInfo` method `AppendSlave` will have to be invoked. This takes the form

```
$TaskInfo AppendSlave <spawn count> <spawn command>
```

where <`spawn command`> is the command to launch the slave process, and <`spawn count`> is the number of slaves to launch with this command. (Typically <`spawn count`> should not be larger than the number of processors on the target system.) The default value for this item (which gets overwritten with the first call to `$TaskInfo AppendSlave`) is

```
 1 {exec %tclsh %oommf batchslave -tk 0 %connect_info batchsolve.tcl}
```

This uses the OOMMF bootstrap program to launch the **batchslave** application, with connection information provided by the master, and using the auxiliary script `batchsolve.tcl`. The `batchmaster` script provides several percent-style substitutions useful in slave launch scripts: %tclsh, %oommf, %connect_info, %oommf_root, and %%. The first is the Tcl shell to use, the second is an absolute path to the OOMMF bootstrap program on the master machine, the third is connection information needed by the `batchslave` application, the fourth is the path to the OOMMF root directory on the master machine, and the last is interpreted as a single percent.

To launch slaves on a remote host, use `rsh` in the spawn command, e.g.,

```
 $TaskInfo AppendSlave 1 {exec rsh foo tclsh oommf/oommf.tcl \
      batchslave -tk 0 %connect_info batchsolve.tcl}
```

This example assumes `tclsh` is in the execution path on the remote machine `foo`, and OOMMF is installed off of your home directory. In addition, you will have to add the machine `foo` to the host connect list with

```
$TaskInfo ModifyHostList +foo
```

and `batchmaster` must be run with the network interface specified as the server host (instead of the default `localhost`), e.g.,

```
tclsh oommf.tcl batchmaster multitask.tcl bar
```

where `bar` is the name of the local machine.

This may seem a bit complicated, but the examples in the next section should make things clearer.

```
# FILE: simpletask.tcl
#
# This is a sample batch task file.  Usage example:
#
#   tclsh oommf.tcl batchmaster simpletask.tcl

# Form task list
$TaskInfo AppendTask A "BatchTaskRun taskA.mif"
$TaskInfo AppendTask B "BatchTaskRun taskB.mif"
$TaskInfo AppendTask C "BatchTaskRun taskC.mif"
```

Figure 2: Sample task script `simpletask.tcl`.

16.2.3 Sample task scripts

The first sample task script (Fig. 2) is a simple example that runs the 3 micromagnetic simulations described by the MIF files `taskA.mif`, `taskB.mif` and `taskC.mif`. It is launched with the command

```
tclsh oommf.tcl batchmaster simpletask.tcl
```

This example uses the default slave launch script, so a single slave is launched on the current machine, and the 3 simulations will be run sequentially. Also, no slave init script is given, so the default procs in `batchsolve.tcl` are used. Output will be magnetization states and tabular data for each equilibrium state, stored in files on the local machine with base names as specified in the MIF files.

The second task script (Fig. 3) is a more complicated example running concurrent processes on two machines. This script should be run with the command

```
tclsh oommf.tcl batchmaster multitask.tcl bar
```

where `bar` is the name of the local machine.

Near the top of the `multitask.tcl` script several Tcl variables (`RMT_MACHINE` through `A_list`) are defined; these are used farther down in the script. The remote machine is specified as `foo`, which is used in the `$TaskInfo AppendSlave` and `$TaskInfo ModifyHostList` commands.

There are two `AppendSlave` commands, one to run two slaves on the local machine, and one to run a single slave on the remote machine (`foo`). The latter changes to a specified working directory before launching the `batchslave` application on the remote machine.

(For this to work you must have rsh configured properly. In the future it may be possible to launch remote commands using the OOMMF account server application, thereby lessening the reliance on system commands like rsh.)

Below this the slave init script is defined. The Tcl regsub command is used to place the task script defined value of BASEMIF into the init script template. The init script is run on the slave when the slave is first brought up. It first reads the base MIF file into a newly created mms_mif instance. (The MIF file needs to be accessible by the slave process, irrespective of which machine it is running on.) Then replacement SolverTaskInit and SolverTaskCleanup procs are defined. The new SolverTaskInit interprets its first argument as a value for the exchange constant A. Note that this is different from the default SolverTaskInit proc, which interprets its first argument as the name of a MIF file to load. With this task script, a MIF file is read once when the slave is brought up, and then each task redefines only the value of A for the simulation (and corresponding changes to the output filenames and data table header).

Finally, the Tcl loop structure

```
foreach A $A_list {
    $TaskInfo AppendTask "A=$A" "BatchTaskRun $A"
}
```

is used to build up a task list consisting of one task for each value of A in A_list (defined at the top of the task script). For example, the first value of A is 10e-13, so the first task will have the label A=10e-13 and the corresponding script is BatchTaskRun 10e-13. The value 10e-13 is passed on by BatchTaskRun to the SolverTaskInit proc, which has been redefined to process this argument as the value for A, as described above.

There are 6 tasks in all, and 3 slave processes, so the first three tasks will run concurrently in the 3 slaves. As each slave finishes it will be given the next task, until all the tasks are complete.

```
# FILE: multitask.tcl
#
# This is a sample batch task file.  Usage example:
#
#   tclsh oommf.tcl batchmaster multitask.tcl hostname [port]
#
# Task script configuration
set RMT_MACHINE   foo
set RMT_TCLSH      tclsh
set RMT_OOMMF      "/path/to/oommf/oommf.tcl"
set RMT_WORK_DIR   "/path/to/oommf/app/mmsolve/data"
set BASEMIF taskA
set A_list { 10e-13 10e-14 10e-15 10e-16 10e-17 10e-18 }

# Slave launch commands
$TaskInfo ModifyHostList +$RMT_MACHINE
$TaskInfo AppendSlave 2 "exec %tclsh %oommf batchslave -tk 0 \
        %connect_info batchsolve.tcl"
$TaskInfo AppendSlave 1 "exec rsh $RMT_MACHINE \
        cd $RMT_WORK_DIR \\\;\
        $RMT_TCLSH $RMT_OOMMF batchslave -tk 0 %connect_info batchsolve.tcl"

# Slave initialization script (with batchsolve.tcl proc
# redefinitions)
set init_script {
    # Initialize solver. This is run at global scope
    set basename __BASEMIF__      ;# Base mif filename (global)
    mms_mif New mif
    $mif Read [FindFile ${basename}.mif]
    # Redefine TaskInit and TaskCleanup proc's
    proc SolverTaskInit { args } {
        global mif outtextfile basename
        set A [lindex $args 0]
        set outbasename "$basename-A$A"
        $mif SetA $A
        $mif SetOutBaseName $outbasename
        set outtextfile [open "$outbasename.odt" "a+"]
        puts $outtextfile [GetTextData header \
                "Run on $basename.mif, with A=[$mif GetA]"]
        flush $outtextfile
    }
    proc SolverTaskCleanup { args } {
        global outtextfile
        close $outtextfile
    }
}
# Substitute $BASEMIF in for __BASEMIF__ in above script
regsub -all -- __BASEMIF__ $init_script $BASEMIF init_script
```

```
$TaskInfo SetSlaveInitScript $init_script

# Create task list
foreach A $A_list {
    $TaskInfo AppendTask "A=$A" "BatchTaskRun $A"
}
```

Figure 3: Advanced sample task script `multitask.tcl`.

17 File Formats

17.1 Problem specification format (MIF)

Micromagnetic simulations are specified to the OOMMF solver using the OOMMF *Micromagnetic Input Format* (MIF). This is also the native file type for the OOMMF problem editor, **mmProbEd** (Sec. 7). All values are in SI units.

A sample MIF file is presented in Fig. 4. The first line of a MIF file must be of the form "`#` MIF x.y", where x.y represents the format revision number. OOMMF reads and writes the MIF 1.1 format. (There was a MIF 1.0 format, but it was never part of a released version of OOMMF.) It is recommended that MIF files be given names ending in the `.mif` file extension so that MIF files may be easily identified.

After the format identifier line, any line ending in a backslash, '\', is joined to the succeeding line before any other processing is performed. Lines beginning with a '`#`' character are comments and are ignored. Blank lines are also ignored.

All other lines must consist of a *Record Identifier* followed by a parameter list. The Record Identifier is separated from the parameter list by one or more ':' and/or '=' characters. Whitespace and case is ignored in the Record Identifier field.

The parameter list must be a proper Tcl list. The parameters are parsed (broken into separate elements) following normal Tcl rules; in short, items are separated by whitespace, except as grouped by double quotes and curly braces. The grouping characters are removed during parsing. Any '`#`' character that is found outside of any grouping mechanism is interpreted as a comment start character. The '`#`' and all following characters on that line are interpreted as a comment.

Order of the records in a MIF file is unimportant, except as explicitly stated below. If two or more lines contain the same Record Identifier, then the last one takes precedence (except for Field Range records, of which there may be several active). All records are required unless listed as optional. Some of these record types are not yet supported by **mmProbEd**, however your may edit a MIF file by hand and supply it to **mmSolve2D** (Sec. 9) using **FileSource** (Sec. 8).

For convenience, the Record Identifier tags are organized into several groups; these groups correspond to the buttons presented by **mmProbEd**. We follow this convention below.

Material parameters

- **# Material Name:** This is a convenience entry for **mmProbEd**; inside the MIF file it is a comment line. It relates a symbolic name (e.g., Iron) to specific values to the next 4 items. Ignored by solvers.

- **Ms:** Saturation magnetization in A/m.
- **A:** Exchange stiffness in J/m.
- **K1:** Crystalline anisotropy constant in J/m^3.
- **Anisotropy Type:** Crystalline anisotropy type; One of < uniaxial | cubic >.
- **Anisotropy Dir1:** Directional cosines of first crystalline anisotropy axis, taken with respect to the coordinate axes (3 numbers). Optional; Default is 1 0 0 (x-axis).
- **Anisotropy Dir2:** Directional cosines of second crystalline anisotropy axis, taken with respect to the coordinate axes (3 numbers). Optional; Default is 0 1 0 (y-axis).

  For uniaxial materials it suffices to specify only Anisotropy Dir1. For cubic materials one should also specify Anisotropy Dir2; the third axis direction will be calculated as the cross product of the first two. The anisotropy directions will be automatically normalized if necessary, so for example 1 1 1 is valid input (it will be modified to .5774 .5774 .5774). For cubic materials, Dir2 will be adjusted to be perpendicular to Dir1 (by subtracting out the component parallel to Dir1).
- **Anisotropy Init:** Method to use to set up directions of anisotropy axes, as a function of spatial location; This is a generalization of the Anisotropy Dir1/2 records. The value for this record should be one of < Constant | UniformXY | UniformS2 >. Constant uses the values specified for Anisotropy Dir1 and Dir2, with no dispersion. UniformXY ignores the values given for Anisotropy Dir1 and Dir2, and randomly varies the anisotropy directions uniformly in the xy-plane. UniformS2 is similar, but randomly varies the anisotropy directions uniformly on the unit sphere (S^2). This record is optional; the default value is Constant.
- **Do Precess:** If 1, then enable the precession term in the Landau-Lifshitz ODE. If 0, then do pure damping only. (Optional; default value is 1.)
- **Gyratio:** The gyromagnetic ratio, in m/(A.s). This is optional, with default value of 2.21×10^5. See the discussion of the Landau-Lifshitz ODE under the Damp Coef record identifier description.
- **Damp Coef:** The ODE solver in OOMMF integrates the Landau-Lifshitz equation, written as

$$\frac{d\mathbf{M}}{dt} = -\gamma\,\mathbf{M} \times \mathbf{H}_{\text{eff}} - \frac{\gamma\alpha}{M_s}\mathbf{M} \times (\mathbf{M} \times \mathbf{H}_{\text{eff}})\,,$$

where

γ	is the gyromagnetic ratio (in m/(A.s)),
α	is the damping coefficient (dimensionless).

The last is specified by the "Damp Coef" entry in the MIF file. If not specified, a default value of 0.5 is used, which allows the solver to converge in a reasonable number of iterations. Physical materials will typically have a damping coefficient in the range 0.004 to 0.15.

Demag specification

- **Demag Type:** Specify algorithm used to calculate self-magnetostatic (demagnetization) field. Must be one of
  - **ConstMag:** Calculates the *average* field in each cell under the assumption that the magnetization is constant in each cell, using formulae from [10]. (The other demag options calculate the field at the center of each cell.)
  - **3dSlab:** Calculate the in-plane field components using offset blocks of constant (volume) charge. Details are given in [2]. Field components parallel to the z-axis are calculated using squares of constant (surface) charge on the upper and lower surfaces of the sample.
  - **3dCharge:** Calculate the in-plane field component using rectangles of constant (surface) charge on each cell. This is equivalent to assuming constant magnetization in each cell. The z-components of the field are calculated in the same manner as for the 3dSlab approach.
  - **FastPipe:** Algorithm suitable for simulations that have infinite extent in the z-direction. This is a 2D version of the 3dSlab algorithm.
  - **None:** No demagnetization. Fastest but least accurate method. `:-}`

  The 3dSlab and 3dCharge methods require that the Part Thickness (cf. the Part Geometry section) be set. All algorithms use Fast Fourier Transform (FFT) techniques to accelerate the calculations.

Part geometry

- **Part Width:** Nominal part width (x-dimension) in meters. Will be automatically adjusted to an integral multiple of Cell Size.

- **Part Height:** Nominal part height (y-dimension) in meters. Will be automatically adjusted to an integral multiple of Cell Size.
- **Part Thickness:** Part thickness (z-dimension) in meters. Required for 3D models.
- **Cell Size:** In-plane (xy-plane) edge dimension of base calculation cell. This cell is a rectangular brick, with square in-plane cross-section and thickness given by Part Thickness.
- **Part Shape:** Optional. Part shape in the xy-plane; must be one of the following:
  - **Rectangle**
    The sample fills the area specified by Part Width and Part Height. (Default.)
  - **Ellipse**
    The sample (or the magnetically active portion thereof) is an ellipse inscribed into the rectangular area specified by Part Width and Part Height.
  - **Oval r**
    Shape is a rounded rectangle, where each corner is replaced by a quarter circle with radius r, where r is the second parameter.
  - **Mask filename**
    Shape is determined by a 2-color bitmap file, the name of which is specified as the second parameter. The given filename must be accessible to the solver application. At present the bitmap file must be in either the PPM (portable pixmap), GIF, or BMP formats. (Any format other than the PPM P3 (text) format may be handled by spawning a `any2ppm` (Sec. 15.2) subprocess.) The bitmap will be scaled as necessary to fit the simulation. The magnetically active cells correspond to black pixels in the bitmap.

Initial magnetization

- **Init Mag:** Name of routine to use to initialize the simulation magnetization directions (as a function of position), and routine parameters, if any. Optional, with default Random. The list of routines is long, and it is easy to add new ones. See the file `maginit.cc` for details. A few of the more useful routines are:
  - **Random**
    Random directions on the unit sphere. This is somewhat like a quenched thermal demagnetized state.

- **Uniform** θ ϕ
  Uniform magnetization in the direction indicated by the two additional parameters, θ and ϕ, where the first is the angle from the z-axis (in degrees), and the second is the angle from the x-axis (in degrees) of the projection onto the xy-plane.
- **Vortex**
  Fits an idealized vortex about the center of the sample.
- **avfFile filename**
  The second parameter specifies an OVF/VIO (i.e., "any" vector field) file to use to initialize the magnetization. The grid in the input file will be scaled as necessary to fit the grid in the current simulation. The file must be accessible to the intended solver application.

Experiment parameters

The following records specify the applied field schedule:

- **Field Range:** Specifies a range of applied fields that are stepped though in a linear manner. The parameter list should be 7 numbers, followed by optional control point (stopping criteria) specifications. The 7 required fields are the begin field Bx By Bz in Tesla, the end field Bx By Bz in Tesla, and an integer number of steps (intervals) to take between the begin and end fields (inclusive). Use as many Field Range records as necessary—they will be stepped through in order of appearance. If the begin field for one range is the same as the end field for the immediately preceding range, then the begin field is not repeated. If the step count is 0, then the end field is ignored and only the begin field is applied. (If the step count is 0, and the begin field is the same as the preceding end field, then the range is skipped.)

  The optional control point specs determine the conditions that cause the applied field to be stepped, or more precisely, ends the simulation of the magnetization evolution for the current applied field. The control point specs are specified as *–type value* pairs. There are 3 recognized control point types: **–torque**, **–time**, and **–iteration**. If a –torque pair is given, then the simulation at the current applied field is ended when $\|\mathbf{m} \times \mathbf{h}\|$ (i.e., $\|\mathbf{M} \times \mathbf{H}\|/M_s^2$) at all spins in the simulation is smaller than the specified –torque value (dimensionless). If a –time pair is given, then the simulation at the current field is ended when the elapsed simulation time *for the current field step* reaches the specified –time value (in seconds). Similarly, an –iteration pair steps the applied field when the iteration count for the current field step reaches the –iteration value. If multiple control point specs are given, then the applied field is advanced when any one of the specs is met. If no control point specs are given on a range line, then the **Default Control Point Spec** is used.

For example, consider the following Field Range line:

```
Field Range: 0 0 0  0.05 0 0  5  -torque 1e-5 -time 1e-9
```

This specifies 6 applied field values, (0,0,0), (0.01,0,0), (0.02,0,0), ..., (0.05,0,0) (in Tesla), with the advancement from one to the next occurring whenever $\|\mathbf{m} \times \mathbf{h}\|$ is smaller than 1e-5 for all spins, or when 1 nanosecond (simulation time) has elapsed at the current field. (Alternately, if –torque was not specified, then the applied field would be stepped at 1, 2, 3, 4 and 5 ns in simulation time.)

This Field Range record is optional, with a default value of 0 0 0 0 0 0 1.

- **Default Control Point Spec:** List of control point *–type value* pairs to use as stepping criteria for any field range with no control point specs. This is a generalization of and replacement for the *Converge |mxh| Value* record. Optional, with default "-torque 1e-5."

- **Field Type:** External field routine and parameters, if any. This is optional, with default Uniform. At most one record of this type is allowed, but the Multi type may be used to apply a collection of fields. The nominal applied field (NAF) is stepped through the Field Ranges described above, and is made available to the external field routines which use or ignore it as appropriate.

  The following Field Type routines are available:

  - **Uniform**
    Applied field is uniform with value specified by the NAF.
  - **Ribbon relcharge x0 y0 x1 y1 height**
    Charge "Ribbon," lying perpendicular to the *xy*-plane. Here relcharge is the charge strength relative to Ms, and (x0,z0), (x1,z1) are the endpoints of the ribbon (in meters). The ribbon extends height/2 above and below the calculation plane. This routine ignores the NAF.
  - **Tie rfx rfy rfz x0 y0 x1 y1 ribwidth**
    The points (x0,y0) and (x1,y1) define (in meters) the endpoints of the center spine of a rectangular ribbon of width ribwidth lying in the *xy*-plane. The cells

with sample point inside this rectangle see an applied field of (rfx,rfy,rfz), in units relative to Ms. (If the field is large, then the magnetizations in the rectangle will be "tied" to the direction of that field.) This routine ignores the NAF.

- **OneFile filename multiplier**
  Read B field in from a file. Each value in the file is multiplied by the "multiplier" value on input. This makes it simple to reverse field direction (use -1 for the multiplier), or to convert H fields to B fields (use 1.256637e-6). The input file may be any of the vector field file types recognized by **mmDisp**. The input dimensions will be scaled as necessary to fit the simulation grid, with zeroth order interpolation as necessary. This routine ignores the NAF.
- **FileSeq filename procname multiplier**
  This is a generalization of the OneFile routine that reads in fields from a sequence of files. Here "filename" is the name of a file containing Tcl code to be sourced during problem initialization, and "procname" is the name of a Tcl procedure defined in filename, which takes the nominal B field components and field step count values as imports (4 values total), and returns the name of the vector field file that should be used as the applied B field for that field step.
- **Multi routinecount **
  **param1count name1 param1 param2 . . . **
  **param2count name2 param1 param2 . . . **
  . . .
  Allows a conglomeration of several field type routines. All entries must be on the same logical line, i.e., end physical lines with '\' continuation characters as necessary. Here routinecount is the number of routines, and param1count is the number parameters (including name1) needed by the first routine, etc.

Note that all lengths are in meters. The coordinates in the simulation lie in the first octant, running from (0,0,0) to (Part Width, Part Height, Part Thickness).

Output specification

- **Base Output Filename:** Default base name used to construct output filenames.

Miscellaneous

- **Converge |mxh| Value:** Nominal value to use as a stopping criterion: When $\|\mathbf{m}\times\mathbf{h}\|$ (i.e., $\|\mathbf{M}\times\mathbf{H}\|/M_s^2$) at all spins in the simulation is smaller than this value, it is assumed that a relaxed (equilibrium) state has been reached for the current applied field. This

is a dimensionless value.
NOTE: This Record Identifier is deprecated. Use *Default Control Point Spec* instead.

- **Randomizer Seed:** Value with which to seed random number generator. Optional. Default value is 0, which uses the system clock to generate a semi-random seed.
- **Max Time Step:** Limit the maximum ODE step size to no larger than this amount, in seconds. Optional.
- **Min Time Step:** Limit the minimum ODE step size to no less than this amount, in seconds. Optional.
- **User Comment:** Free-form comment string that may be used for problem identification. Optional.

```
# MIF 1.1
#
# All units are SI.
#
####################### MATERIAL PARAMETERS ############################
Ms:  800e3                # Saturation magnetization in A/m.
A:   13e-12               # Exchange stiffness in J/m.
K1:  0.5e3                # Anisotropy constant in J/m^3.
Anisotropy Type: uniaxial  # One of <uniaxial|cubic>.
Anisotropy Dir1: 1 0 0     # Directional cosines wrt to coordinate axes

####################### DEMAG SPECIFICATION ############################
Demag Type: ConstMag # One of <ConstMag|3dSlab|2dSlab|3dCharge|FastPipe|None>.

########################## PART GEOMETRY ###############################
Part Width:     0.25e-6    # Nominal part width in m
Part Height:    1.0e-6     # Nominal part height in m
Part Thickness: 1e-9       # Part thickness in m.
Cell Size:      8.1e-9     # Cell size in m.
#Part Shape:    # One of <Rectangle|Ellipse|Oval|Mask>.  Optional.

###################### INITIAL MAGNETIZATION ###########################
Init Mag: Uniform 90 45 # Initial magnetization routine and parameters
```

```
##################### EXPERIMENT PARAMETERS #########################
Field Range: -.05 -.01 0. .05 .01 0. 100 # Start_field Stop_field Steps
Field Range: .05 .01 0. -.05 -.01 0. 100
Field Type: Multi 4 \
 7 Ribbon 1 0 1.0e-6 0.25e-6 1.0e-6 1e-9 \
 7 Ribbon 1 0 0     0.25e-6 0     1e-9 \
 9 Tie 100 0 0 0.12e-6 0.5e-6 0.13e-6 0.5e-6 8.1e-9 \
 1 Uniform
# The above positions ribbons of positive charge along the upper
# and lower edges with strength Ms, applies a large (100 Ms) field
# to the center cell, and also applies a uniform field across the
# sample stepped from (-.05,-.01,0.) to (.05,.01,0.) (Tesla), and
# back, in approximately 0.001 T steps.

Default Control Point Spec: -torque 1e-6 # Assume equilibrium has been
# reached, and step the applied field, when the reduced torque |mxh|
# drops below 1e-6.

##################### OUTPUT SPECIFICATIONS #########################
Base Output Filename: samplerun

######################### MISCELLANEOUS ###############################
Randomizer Seed: 1   # Value to seed random number generator with.
User Comment: This is an example MIF file, with lots of comments.
```

Figure 4: Example MIF file.

17.2 Data table format (ODT)

Textual output from solver applications that is not of the vector field variety is output in the *OOMMF Data Table* (ODT) format. This is an ASCII file format, with column information in the header and one line of data per record. Any line ending in a '\' character is joined to the succeeding line before any other processing is performed. Any leading '#' characters on the second line are removed.

As with the OVF format (Sec. 17.3.1), all non-data lines begin with a '#' character, comments with two '#' characters. (This makes it easier to import the data into external programs, for example, plotting packages.) An example is shown in Fig. 5.

The first line of an ODT file should be the file type descriptor

```
# ODT 1.0
```

It is also recommended that ODT files be given names ending in the file extension `.odt` so that ODT files may be easily identified.

The remaining lines of the ODT file format should be comments, data, or any of the following 5 recognized descriptor tag lines:

- **# Table Start:** Optional, used to segment a file containing multiple data table blocks. Anything after the colon is taken as an optional label for the corresponding data table block.
- **# Title:** Optional; everything after the colon is interpreted as a title for the table.
- **# Columns:** Required. One parameter per column, designating the label for that column. Spaces may be embedded in a column label by using the normal Tcl grouping mechanisms (i.e., double-quotes and braces).
- **# Units:** Optional. If given, it should have one parameter for each column, giving a unit label for the corresponding column.
- **# Table End**: Optional, no parameters. Should be paired with a corresponding Table Start record.

Data may appear anywhere after the Columns descriptor record and before any Table End line, with one record per line. The data should be numeric values separated by whitespace.

17.3 Vector field format (OVF)

Vector field files specify vector quantities (e.g., magnetization or magnetic flux density) as a function of spatial position. This type of file is produced by **mmSolve2D** (Sec. 9) when "Total Field" or "Magnetization" output is selected. It is also the input data type read by **mmDisp** (Sec. 12). OOMMF stores vector field files in the *OOMMF Vector Field* (OVF) format. There are two versions of the OVF format supported by OOMMF. The OVF 1.0 format is the preferred format and the only one written by OOMMF software. It supports both rectangular and irregular meshes, in binary and ASCII. The OVF 0.0 format is an older, simpler format that can be useful for importing vector field data into OOMMF from other programs. (A third format, the *Vector Input/Output* (VIO) format, is used by some

```
# ODT 1.0
# Table Start
# Title: This is a small sample ODT file.
#
## This is a sample comment.  You can put anything you want
## on comment lines.
#
# Columns: Iteration "Applied Field"  {Total Energy}    Mx
# Units:       {}          "mT"           "J/m^3"       "A/m"
              103           50            0.00636      787840
             1000           32            0.00603      781120
            10300        -5000            0.00640     -800e3
# Table End
```

Figure 5: Sample ODT file.

precursors to the OOMMF code. Although OOMMF is able to read the VIO format, its use is deprecated. New programs should not make use of it.)

The recommended file extensions for OVF files are `.omf` for magnetization files, `.ohf` for magnetic field (**H**) files, `.obf` for magnetic flux density (**B**) files, or `.ovf` for generic files.

17.3.1 The OVF 1.0 format

A commented sample OVF 1.0 file is provided in Fig. 6. An OVF file has an ASCII header and trailer, and a data block that may be either ASCII or binary. All non-data lines begin with a '`#`' character; the double '`##`' marks the start of a comment, which continues until the end of the line. There is no line continuation character. Lines starting with a '`#`' but containing only whitespace characters are ignored.

All non-empty non-comment lines in the file header are structured as field+value pairs. The field tag consists of all characters after the initial '`#`' up to the first colon (':') character. Case is ignored, and all space and tab characters are eliminated. The value consists of all characters after the first colon, continuing up to a '`##`' comment designator or the end of the line.

The first line of an OVF file should be a file type identification line, having the form

```
# OOMMF: rectangular mesh v1.0
```

where the value "rectangular mesh v1.0" identifies the meshtype and revision. While the OVF 1.0 format was under development in earlier OOMMF releases, the revision strings `0.99` and `0.0a0` were sometimes recorded on the file type identification line. OOMMF treats all of these as synonyms for `1.0` when reading OVF files.

The remainder of the file is conceptually broken into Segment blocks, and each Segment block is composed of a (Segment) Header block and a Data block. Every block begins with "`#` Begin: <block type>" line, and ends with a corresponding "`#` End: <block type>" line. The number of Segment blocks is specified in the

```
# Segment count: 1
```

line. Currently only 1 segment is allowed. This may be changed in the future to allow for multiple vector fields per file. This is followed by

```
# Begin: Segment
```

to start the first segment.

Segment Header block The Segment Header block start is marked by the line "`#` Begin: Header" and the end by "`#` End: Header". Everything between these lines should be either comments or one of the following file descriptor lines. They are order independent. All are required unless otherwise stated. Numeric values are floating point values unless "integer" is explicitly stated.

- **title:** Long file name or title.
- **desc:** Description line. Use as many as desired. Description lines may be displayed by postprocessing programs, unlike comment lines which are ignored by all automated processing.
- **meshunit:** Fundamental mesh spatial unit, treated as a label. The comment marker '`##`' is not allowed in this label. Example value: "nm".
- **valueunit:** Fundamental field value unit, treated as a label. The comment marker '`##`' is not allowed in this label. Example: "kA/m."
- **valuemultiplier:** File data values are multiplied by this to get true values in units of "valueunit." This simplifies the use of normalized values in the data block.
- **xmin, ymin, zmin, xmax, ymax, zmax:** Six separate lines, specifying the bounding box for the mesh, in units of "meshunit." This may be used by display programs to limit the display area, and may be used for drawing a boundary frame if "boundary" is not specified.

- **boundary:** List of *(x,y,z)* triples specifying the vertices of a boundary frame.
- **ValueRangeMaxMag, ValueRangeMinMag:** The maximum and non-zero minimum field magnitudes in the data block, in the same units as used in the data block. These are for optional use as hints by postprocessing programs; for example, **mmDisp** will not display any vector with magnitude smaller than ValueRangeMinMag.
- **meshtype:** Grid structure; should be either "rectangular" or "irregular." Irregular grid files should specify "pointcount" in the header; rectangular grid files should specify instead "xbase, ybase, zbase," "xstepsize, ystepsize, zstepsize," and "xnodes, ynodes, znodes."
- **pointcount:** Number of data sample points/locations, i.e., nodes (integer). For irregular grids only.
- **xbase, ybase, zbase:** Three separate lines, denoting the position of the first point in the data section, in units of "meshunit." For rectangular grids only.
- **xstepsize, ystepsize, zstepsize:** Three separate lines, specifying the distance between adjacent grid points, in units of "meshunit." For rectangular grids only.
- **xnodes, ynodes, znodes:** Three separate lines, specifying the number of nodes along each axis (integers only). For rectangular grids only.

Data block The data block start is marked by a line of the form

```
# Begin: data <representation>
```

where <representation> is one of "text", "binary 4", or "binary 8". Text mode uses the ASCII specification, with individual data items separated by an arbitrary amount of whitespace (spaces, tabs and newlines). Comments are not allowed inside binary mode data blocks, but are permitted inside text data blocks.

The binary representations are IEEE floating point in network byte order (MSB). To insure that the byte order is correct, and to provide a partial check that the file hasn't been sent through a non 8-bit clean channel, the first datum is a predefined value: 1234567.0 (Hex: 49 96 B4 38) for 4-byte mode, and 123456789012345.0 (Hex: 42 DC 12 21 83 77 DE 40) for 8-byte mode. The data immediately follow the check value.

The structure of the data depends on whether the "meshtype" declared in the header is "irregular" or "rectangular". For irregular meshes, each data element is a 6-tuple, consisting of the x, y and z components of the node position, followed by the x, y and z components of the field at that position. Ordering among the nodes is not relevant. The number of nodes is specified in the "pointcount" line in the segment header.

For rectangular meshes, data input is field values only, in x, y, z component triples. These are ordered with the x index incremented first, then the y index, and the z index last. This is nominally Fortran order, and is adopted here because commonly x will be the longest dimension, and z the shortest, so this order is more memory-access efficient than the normal C array indexing of z, y, x. The size of each dimension is specified in the "xnodes, ynodes, znodes" lines in the segment header.

In any case, the first character after the last data item should be a newline, followed by

```
# End: data <representation>
```

where <representation> must match the value in the "Begin: data" line. This is followed by a

```
# End: segment
```

line that ends the segment, and hence the file.

Note: An OVF file with ASCII data using irregular mesh output is also a valid SVF file, although one must pay close attention to possible value scaling as specified by "`# valueunit`" and "`# valuemultiplier`" header lines.

```
# OOMMF: rectangular mesh v1.0
#
## This is a comment.
## No comments allowed in the first line.
#
# Segment count: 1   ## Number of segments.  Should be 1 for now.
#
# Begin: Segment
# Begin: Header
#
# Title: Long file name or title goes here
#
# Desc: 'Description' tag, which may be used or ignored by postprocessing
# Desc: programs. You can put anything you want here, and can have as many
# Desc: 'Desc' lines as you want.  The ## comment marker is disabled in
# Desc: description lines.
#
```

```
## Fundamental mesh measurement unit.  Treated as a label:
# meshunit: nm
#
# meshtype: rectangular
# xbase: 0.      ## (xbase,ybase,zbase) is the position, in
# ybase: 0.      ## 'meshunit', of the first point in the data
# zbase: 0.      ## section (below).
#
# xstepsize: 20. ## Distance between adjacent grid pts.: on the x-axis,
# ystepsize: 10. ## 20 nm, etc.  The sign on this value determines the
# zstepsize: 10. ## grid orientation relative to (xbase,ybase,zbase).
#
# xnodes: 200    ## Number of nodes along the x-axis, etc. (integers)
# ynodes: 400
# znodes:   1
#
# xmin:    0.    ## Corner points defining mesh bounding box in
# ymin:    0.    ## 'meshunit'.  Floating point values.
# zmin:  -10.
# xmax: 4000.
# ymax: 4000.
# zmax:   10.
#
## Fundamental field value unit, treated as a label:
# valueunit: kA/m
# valuemultiplier: 0.79577472  ## Multiply file values by this to get
#                              ## true value in 'valueunits'.
#
# ValueRangeMaxMag:  1005.3096  ## These are in file value units, and
# ValueRangeMinMag:  1e-8       ## are used as hints (or defaults) by
#     ## postprocessing programs.  The mmDisp program ignores any points
#     ## with magnitude smaller than ValueRangeMinMag, and uses
#     ## ValueRangeMaxMag to scale inputs for display.
#
# End: Header
#
## Anything between '# End: Header' and '# Begin: data text',
## '# Begin: data binary 4' or '# Begin: data binary 8' is ignored.
```

```
##
## Data input is in 'x-component y-component z-component' triples,
## ordered with x incremented first, then y, and finally z.
#
# Begin: data text
1000 0 0 724.1 0. 700.023
578.5 500.4 -652.36
<...data omitted for brevity...>
252.34 -696.42 -671.81
# End: data text
# End: segment
```

Figure 6: Commented OVF sample file.

17.3.2 The OVF 0.0 format

The OVF 0.0 format is a simple ASCII format supporting irregularly sampled data. It is intended as an aid for importing data from non-OOMMF programs, and is backwards compatible with the format used for problem submissions for the first μMag standard problem[8].

Users of previous releases of OOMMF may recognize the OVF 0.0 format by its previous name, the Simple Vector Field (SVF) format. It came to the attention of the OOMMF developers that the file extension `.svf` was already registered in several MIME systems to indicate the Simple Vector Format[9], a vector graphics format. To avoid conflict, we have stopped using the name Simple Vector Field format, although OOMMF software still recognizes and reads SVF files, and you may still find example files and other references to the SVF format.

A sample OVF 0.0 file is shown in Fig. 7. Any line beginning with a '`#`' character is a comment, all others are data lines. Each data line is a whitespace separated list of 6 elements: the x, y and z components of a node position, followed by the x, y and z components of the field at that position. Input continues until the end of the file is reached.

It is recommended (but not required) that the first line of an OVF file be

```
# OOMMF: irregular mesh v0.0
```

[8]http://www.ctcms.nist.gov/~rdm/stdprob_1.html
[9]http://www.softsource.com/svf/

```
# OOMMF irregular mesh v0.0
## File: sample.ovf
## Boundary-XY: 0.0 0.0 1.0 0.0 1.0 2.0 0.0 2.0 0.0 0.0
## Grid step: .25 .5 0
#  x      y      z        m_x      m_y      m_z
  0.01   0.01   0.01   -0.35537  0.93472 -0.00000
  0.01   1.00   0.01   -0.18936  0.98191 -0.00000
  0.01   1.99   0.01   -0.08112  0.99670 -0.00000
  0.50   0.50   0.01   -0.03302  0.99945 -0.00001
  0.99   0.05   0.01   -0.08141  0.99668 -0.00001
  0.75   1.50   0.01   -0.18981  0.98182 -0.00000
  0.99   1.99   0.01   -0.35652  0.93429 -0.00000
```

Figure 7: Example OVF 0.0 file.

This will aid automatic file type detection. Also, three special (extended) comments in OVF 0.0 files are recognized by **mmDisp**:

```
## File: <filename or extended filename>
## Boundary-XY: <boundary vertex pairs>
## Grid step: <cell dimension triple>
```

All these lines are optional. The "File" provides a preferred (possibly extended) filename to use for display identification. The "Boundary-XY" line specifies the ordered vertices of a bounding polygon in the xy-plane. If given, **mmDisp** will draw a frame using those points to ostensibly indicate the edges of the simulation body. Lastly, the "Grid step" line provides three values representing the average x, y and z dimensions of the volume corresponding to an individual node (field sample). It is used by **mmDisp** to help scale the display.

Note that the data section of an OVF 0.0 file takes the simple form of columns of ASCII formatted numbers. Columns of whitespace separated numbers expressed in ASCII are easy to import into other programs that process numerical datasets, and are easy to generate, so the OVF 0.0 file format is useful for exchanging vector field data between OOMMF and non-OOMMF programs. Furthermore, the data section of an OVF 0.0 file is consistent with the data section of an OVF 1.0 file that has been saved as an irregular mesh using text data representation. This means that even though OOMMF software now writes only the OVF 1.0 format for vector field data, simple interchange of vector field data with other programs is still supported.

18 Troubleshooting

The OOMMF developers rely on reports from OOMMF users to alert them to problems with the software and its documentation, and to guide the selection and implementation of new features. See the Credits (Sec. 20) for instructions on how to contact the OOMMF developers.

The more complete your report, the fewer followup messages will be required to determine the cause of your problem. Usually when a problem arises there is an error message produced by the OOMMF software. A stack trace may be offered that reveals more detail about the error. When reporting an error, it will help the developers diagnose the problem if users cut and paste into their problem report the error message and stack trace exactly as reported by OOMMF software. In addition, please include a copy of the output generated by `tclsh oommf.tcl -platform` so that the OOMMF developers will know the details of your platform configuration.

19 References

[1] A. Aharoni, *Introduction to the Theory of Ferromagnetism* (Oxford, New York, 1996).

[2] D. V. Berkov, K. Ramstöck and A. Hubert, "Solving Micromagnetic Problems: Towards an Optimal Numerical Method," *Phys. Stat. Sol. (a),* **137**, 207–222 (1993).

[3] W. F. Brown Jr., *Micromagnetics* (Krieger, New York, 1978).

[4] M. J. Donahue and R. D. McMichael, "Exchange Energy Representations in Computational Micromagnetics," *Physica B,* **233**, 272–278 (1997).

[5] T. L. Gilbert, "A Lagrangian Formulation of the Gyromagnetic Equation of the Magnetization Field," *Phys. Rev.,* **100**, 1243 (1955).

[6] P. R. Gillette and K. Oshima, "Magnetization Reversal by Rotation," *J. Appl. Phys.,* **29** 529-531 (1958).

[7] L. Landau and E. Lifshitz, "On the Theory of the Dispersion of Magnetic Permeability in Ferromagnetic Bodies, *Physik. Z. Sowjetunion,* **8**, 153–169 (1935).

[8] R. D. McMichael and M. J. Donahue, "Head to Head Domain Wall Structures in Thin Magnetic Strips," *IEEE Trans. Mag.,* **33**, 4167–4169 (1997).

[9] L. Néel, "Some Theoretical Aspects of Rock-Magnetism," *P. M. Suppl.,* 191 (1955).

[10] A. J. Newell, W. Williams and D. J. Dunlop, "A Generalization of the Demagnetizing Tensor for Nonuniform Magnetization," *J. Geophysical Research,* **98**, 9551–9555 (1993).

[11] M. R. Scheinfein, J. Unguris, J. L. Blue, K. J. Coakley, D. T. Pierce and R. J. Celotta, "Micromagnetics of Domain Walls at Surfaces," *Phys. Rev. B,* **43**, 3395–3422 (1991).

[12] E. C. Stoner and E. P. Wohlfarth, "A Mechanism of Magnetic Hysteresis in Heterogeneous Alloys," *Phil. Trans. Royal Soc. London*, **A240**, 599–642 (1948).

20 Credits

The main contributors to this document are Michael J. Donahue (michael.donahue@nist.gov) and Donald G. Porter (donald.porter@nist.gov), both of ITL/NIST. Section 3 is based on notes from Dianne P. O'Leary (oleary@cs.umd.edu).

The OOMMF[10] code is being developed mainly by Michael Donahue and Donald Porter. Robert D. McMichael (rmcmichael@nist.gov) made contributions to the early development of the 2D micromagnetic solver. Jason Eicke (jeicke@seas.gwu.edu) is responsible for the problem editor, and has worked on the self-magnetostatic module of the micromagnetic solver.

Quite a few users have contributed to the development of OOMMF by submitting bug reports, small pieces of code, or suggestions for improvements. Many thanks to all these people, including Olivier Gerardin, Ping He, Michael Ho, Michael Kleiber, H. T. Leung, David Lewis, Sang Ho Lim, Van Luu, Andy P. Manners, Anil Prabhakar, Robert Ravlic, Stephen E. Russek, Renat Sabirianov, Zhupei Shi, Xiaobo Tan, Stephen Thompson, Pieter Visscher, Scott L. Whittenburg, Kong Xiangyang, Chengtao Yu, Steven A. Zielke, and Pei Zou.

If you have bug reports, contributed code, feature requests, or other comments for the OOMMF developers, please send them in an e-mail message to `<michael.donahue@nist.gov>`.

[10] http://math.nist.gov/oommf/

This page intentionally blank.