

Research software engineering (Ubermag)



Hans Fangohr

SSU Computational Science

Max Planck Institute for the Structure and Dynamics of Matter

Hamburg, Germany

and

University of Southampton, UK

2022-05-12

hans.fangohr@mpsd.mpg.de

[@ProfCompMod](#) 

Introduction Ubermag

Motivation for Ubermag

Research Software Engineering practices

Ubermag-specific aspects

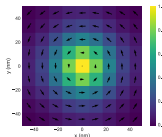
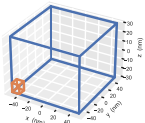
Documentation and Jupyter notebooks

Summary

Introduction Ubermag



$$E = \int_V [-\mathbf{A}\mathbf{m} \cdot \nabla^2 \mathbf{m} - \mu_0 M_s \mathbf{m} \cdot \mathbf{H} + w_d] dV$$



```
In [1]: system = mm.System(name='vortex')
system.energy = mm.Exchange(A=1e-11) + mm.Demag()
system.dynamics = (mm.Precession(c.gamma0)
+ mm.Damping(alpha=0.2))
```

```
In [2]: mesh = df.Mesh(p1=(-50e-9, -50e-9, -30e-9),
p2=(50e-9, 50e-9, 30e-9),
cell=(10e-9, 10e-9, 10e-9))
system.m = df.Field(mesh, dim=3, norm=4e5,
value=vortex_fun)
```

```
In [3]: md.drive(system)
```

```
In [4]: system.m.orientation.plane('z').mpl()
```



Automated
backend



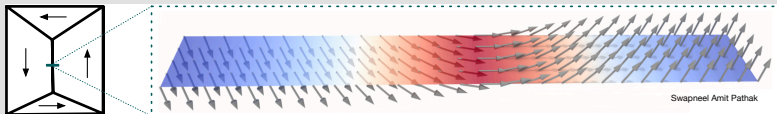
- Home page: <https://ubermag.github.io>,
- Source: <https://github.com/ubermag>
- Beg et.al., *Ubermag: Toward More Effective Micromagnetic Workflows*, 10.1109/TMAG.2021.3078896 (2022)

- magnetism at the nanometre to micrometre length scale
- physical property of interest: magnetisation vector field \mathbf{m}

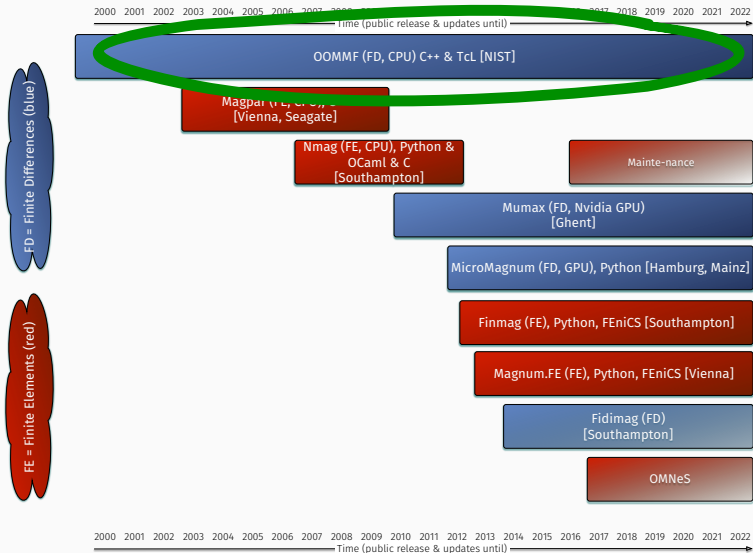
$$\mathbf{m}: \mathbb{R}^3 \rightarrow \mathbb{R}^3$$
$$\mathbf{r} \mapsto \mathbf{m}(\mathbf{r})$$

Example:

- domain formation in ferromagnets (left)
- details of domain walls are resolved (right)



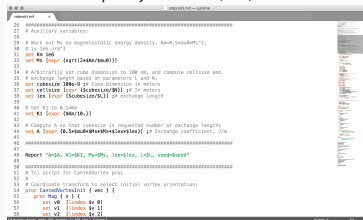
Existing micromagnetic simulation packages



OOMMF: Object Oriented Micromagnetic framework

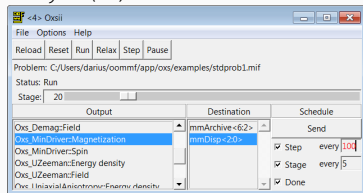
- Probably the most widely used micromagnetic simulation tool
- Developed at National Institute for Standards and Technology (NIST), US, since ~1998 (M. Donahue & D. Porter)
- Cited over 3300 times in scientific publications
- Written in C++ & Tcl
- <https://math.nist.gov/oommf/>

Problem specification (Tcl):



```
#####
25 #####
26 # Auxiliary variables:
27 #
28 # Work set M0 as magnetostatic energy density,  $M_0 = 4\pi M_0^2$ .
29 #  $\mu_0$  is  $4\pi \times 10^{-7}$ 
30 set mu0 [expr {4.0*3.14159265359*10^-7}]
31 set M0 [expr {sqrt(2*600*mu0/3)}]
32
33 #
34 # Arbitrarily set cube dimension to 100 nm, and compute cellsize and
35 # exchange length based on parameters L and H.
36 set cubeSize 100.0 # cube dimension in meters
37 set cellSize [expr {($cubeSize/100)}] # in meters
38 set tex [expr {($cubeSize/10)}] # exchange length
39
40 # Set K1 to 0.3e6
41 set K1 [expr {300/10.0}]
42
43 # Compute A to that cubeSize is requested number of exchange lengths
44 set A [expr {($cubeSize/100)*($cellSize/10)}] # exchange coefficient,  $\mu_0$ 
45
46 #####
47 # Report "M0, K1, K2, M0, M0, tex=10x, L=5L, seed=seed"
48
49 #####
50 # Tcl script for ContourPlot proc
51
52 # Coordinate transform to select initial vortex orientation:
53 proc ContourPlotInit {vec} {
54     set Mag { $vec }
55     set v0 [lindex $vec 0]
56     set v1 [lindex $vec 1]
57     set v2 [lindex $vec 2]
58 }
```

GUI for interactive control and data analysis (Tk):

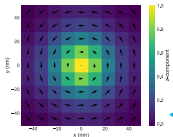
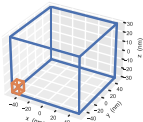


Ubermag is interface to OOMMF



Researcher

$$E = \int_V [-\mathbf{A}\mathbf{m} \cdot \nabla^2 \mathbf{m} - \mu_0 M_s \mathbf{m} \cdot \mathbf{H} + w_d] dV$$



Automated backend

```
In [1]: system = mm.System(name='vortex')
system.energy = mm.Exchange(A=1e-11) + mm.Demag()
system.dynamics = (mm.Precession(c.gamma0)
                  + mm.Damping(alpha=0.2))
```

```
In [2]: mesh = df.Mesh(p1=(-50e-9, -50e-9, -30e-9),
                       p2=(50e-9, 50e-9, 30e-9),
                       cell=(10e-9, 10e-9, 10e-9))
system.m = df.Field(mesh, dim=3, norm=4e5,
                   value=vortex_fun)
```

```
In [3]: md.drive(system)
```

```
In [4]: system.m.orientation.plane('z').mpl()
```



Ubermag team:

Martin Lang, Marjan Beg, Sam Holt, Swapneel Pathak, Hans Fangohr

Motivation for Ubermag

Computational research workflow (Example: OOMMF)

1. Decide what physics problem needs to be solved.
2. Translate physics into OOMMF syntax
3. Run OOMMF simulation to compute results
4. Read simulation output files
5. Analyse and visualise results. Might to go back to 1.
6. Summarise and write up results (in paper)

Ubermag: *Simplify (automate) steps 2 to 4:*

- a) remove OOMMF-specific syntax
- b) ability to use other micromagnetic packages as computational backends

Ubermag for better reproducibility

Ingredients:

Protocol

- Drive simulation through Jupyter notebooks
- Notebooks provide protocol

Software

- Archive which version of the software is used (e.g. ubermag 0.62)

Data

- Keep input data in (git) repository (such as notebooks, additional data files & scripts)

Research Software Engineering practices

Open source and version control

- use of version control
- use permissive license (such as BSD or MIT)
- host code publicly on Github (<https://github.com/ubermag>)



master ▾

ubermag / LICENSE



ubermag/ubermag is licensed under the
BSD 3-Clause "New" or "Revised" License

A permissive license similar to the BSD 2-Clause License, but with a 3rd clause that prohibits others from using the name of the project or its contributors to promote derived products without written consent.

Permissions

- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Private use

Limitations

- ✗ Liability
- ✗ Warranty

This is not legal advice. [Learn more about repository licenses.](#)

Testing: have automated tests

- Idea: in addition to production code, write test functions that test the production code.
- Unit tests, system tests, regression tests
- use `py.test` to collect and execute tests:

```
...
micromagneticmodel/test_energy.py::Energy::test_init PASSED [ 50%]
micromagneticmodel/test_energy.py::Energy::test_init_invalid_args PASSED [ 50%]
micromagneticmodel/test_energy.py::Energy::test_add_sub PASSED [ 50%]
micromagneticmodel/test_energy.py::Energy::test_repr PASSED [ 51%]
micromagneticmodel/test_energy.py::Energy::test_repr_latex PASSED [ 51%]
...
=== 8 failed, 423 passed, 4 skipped, 92 warnings in 1212.06s (0:20:12) =====
```

- make tests executable from within each package:
`import ubermag; ubermag.test()`

Continuous Integration (CI)

- Idea: *automatically execute* all tests for every code change
- Implementation:
 - Github actions for CI
 - run test suites for multiple operating systems

The screenshot shows a GitHub Actions workflow run page. At the top, it displays a green checkmark and the text "workflow workflow #374". To the right, there are buttons for "Re-run all jobs" and a menu icon. Below this, a "Summary" tab is selected, showing a table with the following information:

Triggered via schedule 6 days ago	Status	Total duration	Artifacts
marijanbeg 6233264	Success	23m 3s	-

Below the summary, a list of jobs is shown, all with green checkmarks:

- workflow (ubuntu-latest)
- workflow (macos-latest)
- workflow (windows-latest)

The main content area shows the workflow file "workflow.yml" triggered on a schedule. A "Matrix: workflow" section lists the jobs and their durations:

- workflow (ubuntu-latest) 9m 46s
- workflow (macos-latest) 21m 48s
- workflow (windows-lat... 22m 32s

Software review process

- aim to have another person in the team review a pull request before it is merged
- improves quality
- shares knowledge
- may seem time demanding (but the time might be needed later anyway)
- can slow down development, in particular if team is small and reviews take a long time.

- use <https://pre-commit.com> to run some checks before any `git` commit:
 - `black` (uncompromising formatter, PEP8, also for Jupyter notebooks)
 - `flake8` (additional Python code checks beyond black's capabilities)
 - `isort` (order of import statements)
 - detect remaining merge conflicts in files
 - ...syntax of yaml/toml files
- Advantages
 - time saving as problems that can be detected automatically are detected automatically
 - identical code style from all authors
 - more efficient code review

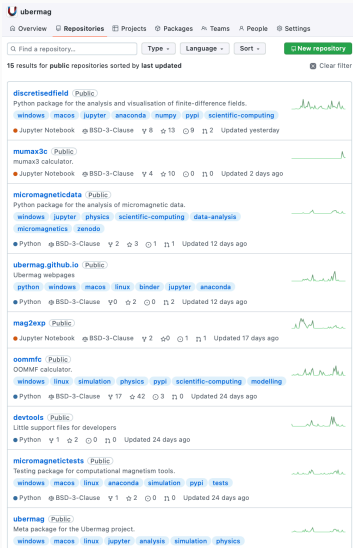
Split project into multiple re-usable python packages

Advantages:

- one (Python) package - one purpose:
 - keeps complexity low
 - keeps each package small(er)
- makes re-use of individual packages more likely

Disadvantages:

- more effort to maintain multiple packages (conda-packages, PyPI entries)



The screenshot shows the Ubermag GitHub repository page. The navigation bar includes links for Overview, Repositories, Projects, Packages, Teams, People, and Settings. A search bar is present with the text "Find a repository...". The page displays 15 results for public repositories, sorted by last updated. The repositories listed are:

- discretisedfield** (Public): Python package for the analysis and visualisation of finite-difference fields. Tags: windows, macos, jupyter, anaconda, numpy, pyqi, scientific-computing. Updated yesterday.
- mumax3c** (Public): mumax3 calculator. Tags: jupyter notebook, BSD-3-Clause. Updated 2 days ago.
- micromagneticdata** (Public): Python package for the analysis of micromagnetic data. Tags: windows, jupyter, physics, scientific-computing, data-analysis, micromagnetics, zenodo. Updated 12 days ago.
- ubermag.github.io** (Public): Ubermag webpages. Tags: python, windows, macos, linux, binder, jupyter, anaconda. Updated 12 days ago.
- mag2exp** (Public): Jupyter Notebook. Tags: BSD-3-Clause. Updated 17 days ago.
- oommf** (Public): OOMMF calculator. Tags: windows, linux, simulation, physics, pyqi, scientific-computing, modelling. Updated 24 days ago.
- devtools** (Public): Little support files for developers. Updated 24 days ago.
- micromagnetictests** (Public): Testing package for computational magnetism tools. Tags: windows, macos, linux, anaconda, simulation, pyqi, tests. Updated 24 days ago.
- ubermag** (Public): Meta package for the Ubermag project. Tags: windows, macos, linux, jupyter, analysis, simulation, physics.

Installation options

- Ubermag:
 - `pip install ubermag` (Python only, needs separate install of OOMMF)
 - `conda install -c conda-forge ubermag` (contains OOMMF)
- OOMMF:
 - install binary package (Windows) from NIST
 - compile from source (Linux, OSX) from NIST
 - conda (`conda install -c conda-forge oommf`)
 - Docker (`docker pull oommf/oommf`)
 - Spack (`spack install oommf`)

Ubermag-specific aspects

Domain-specific language for *micromagnetic problems*

- express physics problem through Python library `micromagneticmodel`
- Cannot *solve* the problem (but is *machine readable*)

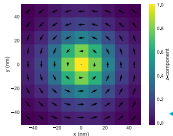
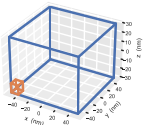
Delegate numerical solution to micromagnetic *calculators*

- given material parameters, geometry and discretisation
- solve numeric version of physics problem through existing simulation packages
 - OOMMF (`oommfc`) / mumax3 (`mumax3c`) / ...?

Ubermag overview



$$E = \int_V [-A\mathbf{m} \cdot \nabla^2 \mathbf{m} - \mu_0 M_s \mathbf{m} \cdot \mathbf{H} + w_d] dV$$



```
In [1]: system = mm.System(name='vortex')
system.energy = mm.Exchange(A=1e-11) + mm.Demag()
system.dynamics = (mm.Precession(c.gamma0)
                  + mm.Damping(alpha=0.2))
```

```
In [2]: mesh = df.Mesh(p1=(-50e-9, -50e-9, -30e-9),
                       p2=(50e-9, 50e-9, 30e-9),
                       cell=(10e-9, 10e-9, 10e-9))
system.m = df.Field(mesh, dim=3, norm=4e5,
                    value=vortex_fun)
```

```
In [3]: md.drive(system)
```

```
In [4]: system.m.orientation.plane('z').mpl()
```



Automated
backend



Embed domain specific language in "general purpose" language

- provide *research environment*:
- user can write arbitrary (Python) program to drive computation and analysis

Connect to existing tools and libraries where possible

- numpy, pandas, xarray, matplotlib, holoviews, ...

Documentation and Jupyter notebooks

Installation Demo Getting **Examples** Package API Changelog Help
started documentation

Search the docs ...

- Standard problem 3
- Standard problem 4
- Standard problem 5**
- FMR standard problem
- Deriving energy values
- Calculating a stray field using an airbox method
- Skyrmion in a disk
- Field operations 2
- Simulation at finite temperature
- Fixed subregions
- Hysteresis simulations
- Multiple energy terms of the same class
- Negative exchange energy constant
- Periodic boundary conditions
- RKKY energy term
- Sine-hysteresis
- Both spatially and time varying field
- Spatially varying parameters 1
- Spatially varying parameters 2
- Time-varying field
- Time-dependent fields and currents

Spin-polarised current

In the second part of simulation, we need to specify the dynamics equation for the system.

```
[9]: system.dynamics += mm.Precession(gamma0=gamma0) + mm.Damping(alpha=alpha) + mm.ZhangLi(u=ux, beta=beta)
system.dynamics
```

$$[9]: -\frac{\gamma_0}{1+\alpha^2} \mathbf{m} \times \mathbf{H}_{\text{eff}} - \frac{\gamma_0 \alpha}{1+\alpha^2} \mathbf{m} \times (\mathbf{m} \times \mathbf{H}_{\text{eff}}) - (\mathbf{u} \cdot \nabla) \mathbf{m} + \beta \mathbf{m} \times [(\mathbf{u} \cdot \nabla) \mathbf{m}]$$

Now, we can drive the system for 8 ns and save the magnetisation in $n = 100$ steps.

```
[10]: td = oc.TimeDriver()
td.drive(system, t=8e-9, n=100)
```

Running OOMMF (ExeOOMMFRunner) [2022/02/25 18:03]... (13.1 s)

The vortex after 8 ns is now displaced from the centre.

```
[11]: system.m.plane(z=0).mpl()
```

- We use notebooks for demo, documentation, tutorials, examples, ...
- Easy to create: just run the simulation in the notebook
- Easy to update
- Sphinx accepts notebook as source for html/pdf documentation
- Can *test* documentation automatically (**nbval**)
- Can have *interactive* documentation (**Binder**)

Testing: Notebooks with NBVAL

```
>> $ py.test --nbval docs/*.ipynb [±master ✓]
===== test session starts =====
platform darwin -- Python 3.9.1, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: /Users/fangohr/git/ubermag-devtools/repos/discretisedfield
plugins: anyio-3.5.0, nbval-0.9.6, cov-3.0.0
collected 434 items

docs/field-definition.ipynb ..... [ 3%]
docs/field-k3d-visualisation.ipynb ..... [ 7%]
docs/field-matplotlib-visualisation.ipynb ..... [ 20%]
docs/field-normalisation.ipynb ..... [ 23%]
docs/field-operations.ipynb ..... [ 41%]
... [ 42%]
docs/field-read-write.ipynb ..... [ 45%]
docs/field-rotations.ipynb ..... [ 50%]
docs/field-spatially-varying.ipynb ..... [ 58%]
docs/field-tools.ipynb ..... [ 62%]
docs/line.ipynb ..... [ 66%]
docs/mesh-basics.ipynb ..... [ 72%]
docs/mesh-bc.ipynb ..... [ 74%]
docs/mesh-line-plane.ipynb ..... [ 77%]
docs/mesh-pad.ipynb ..... [ 79%]
docs/mesh-subregions.ipynb ..... [ 84%]
docs/mesh-visualisation.ipynb ..... [ 87%]
docs/mesh-widgets.ipynb ..... [ 89%]
docs/region-basics.ipynb ..... [ 93%]
docs/region-facing-surfaces.ipynb ... [ 94%]
docs/region-visualisation.ipynb ..... [ 97%]
docs/xarray-usage.ipynb ..... [100%]

===== 434 passed in 162.95s (0:02:42) =====
```

Testing the documentation (Notebooks with NBVAL)

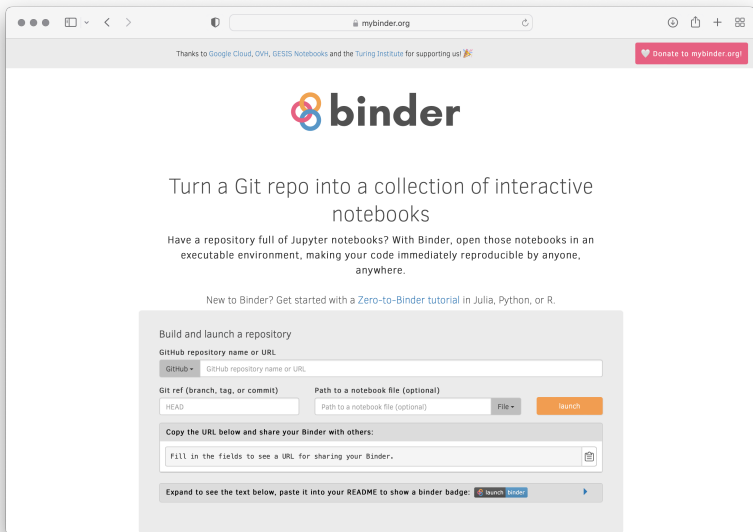
NBVAL to *VAL*idate NoteBooks

- each cell is a test
- test passes if output in file agrees with re-computed output
- **nbval** is extension to **py.test**
<https://github.com/computationalmodelling/nbval>

- Make notebook tests part of the continuous integration
 - notice out-of-date documentation automatically
- get additional system tests "for free"

Fangohr et.al., *Testing with Jupyter notebooks: NBVAL*, arxiv.org/abs/2001.04808 (2021)

Interactive notebooks with Binder



Thanks to Google Cloud, OVH, GESIS Notebooks and the Turing Institute for supporting us! [Donate to mybinder.org!](#)

binder

Turn a Git repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

New to Binder? Get started with a [Zero-to-Binder tutorial](#) in Julia, Python, or R.

Build and launch a repository


GitHub repository name or URL



GitHub

Git ref (branch, tag, or commit) Path to a notebook file (optional)

HEAD File

Copy the URL below and share your Binder with others:



Expand to see the text below, paste it into your README to show a binder badge:  [launch binder](#) 

Example:

`https://ubermag.github.io` → "Try in your browser"

Use-cases for binder-enabled notebook repository:

- interactive documentation
- zero-install test-drive of software (in browser)
 - useful for workshops
- Supports reproducibility and re-usability
- Supports science communication and transparency

Summary

Recommend practice

- version control, automatic tests, continuous integration
- code reviews, pre-commits tools (code style)
- connect to, re-use and build on existing tools
- enable use through Jupyter notebooks:
 - for documentation in notebooks
 - testing of documentation (nbval)
 - interactive documentation (Binder)
 - better reproducibility
- open source
- encourage community contributions

Acknowledgments

This work was supported by the Horizon 2020 OpenDreamKit Project (#676541) and the EPSRC Programme grant on Skyrmionics (#EP/N032128/1).

Contact: hans.fangohr@mpsd.mpg.de, [🐦 @ProfCompMod](https://twitter.com/ProfCompMod)

- Marijan Beg, Martin Lang and Hans Fangohr, *Ubermag: Toward More Effective Micromagnetic Workflows*, in IEEE Transactions on Magnetics, vol. 58, no. 2, pp. 1-5, Feb. 2022, Art no. 7300205, <https://doi.org/10.1109/TMAG.2021.3078896> (2022)
- Marijan Beg, Juliette Belin, Thomas Kluyver, Alexander Konovalov, Min Ragan-Kelley, Nicolas Thiery, Hans Fangohr *Using Jupyter for reproducible scientific workflows*, Computing in Science & Engineering 23, 36-46, <https://doi.org/10.1109/MCSE.2021.3052101> (2021)
- Hans Fangohr, Vidar Fauske, etal, *Testing with Jupyter notebooks: Notebook VALidation (nbval) plug-in for pytest* <https://arxiv.org/abs/2001.04808> (2021)
- Hans Fangohr, Marijan Beg, etal, *Data exploration and analysis with Jupyter notebooks*, Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS2019, TUCPR02, <https://jacow.org/icalepcs2019/papers/tucpr02.pdf> (2020)
- Marijan Beg, Ryan A. Pepper, Hans Fangohr, *User interfaces for computational science: a domain specific language for OOMMF embedded in Python*, AIP Advances 7, 056025, <https://doi.org/10.1063/1.4977225> (2017)

Challenges

- described workflow with Jupyter notebooks works best if simulation time is short
- users misunderstand role of software provider with support desk, micromagnetics expert, tutor, advisor, research collaborator
- what versioning to use (for each package / across packages?)
- duplication of work (for example `discretisedfield` / `xarray`)

Ubermag is interface to OOMMF – longer example



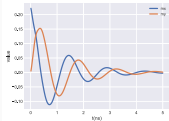
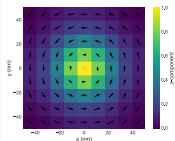
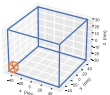
Researcher



Automated backend

$$E = \int_V [-Am \cdot \nabla^2 m - \mu_0 M_s m \cdot \mathbf{H} + w_d] dV$$

$$\frac{\partial \mathbf{m}}{\partial t} = -\frac{\gamma_0}{1 + \alpha^2} \mathbf{m} \times \mathbf{H}_{\text{eff}} - \frac{\gamma_0 \alpha}{1 + \alpha^2} \mathbf{m} \times (\mathbf{m} \times \mathbf{H}_{\text{eff}})$$



Ubermag example: Vortex dynamics jupyter

```
In [1]: import discretisedfield as df
import micromagneticmodel as mm
import oomfmc as mc

In [2]: system = mm.System(name='vortex_dynamics')
system.energy = mm.Exchange(A=13e-12) + mm.Demag()
system.dynamics = (mm.Precession(gamma0=mm.consts.gamma0)
+ mm.Damping(alpha=0.2))

In [3]: region = df.Region(p1=(-50e-9, -50e-9, -30e-9),
p2=(50e-9, 50e-9, 30e-9))
mesh = df.Mesh(region=region,
cell=(10e-9, 10e-9, 10e-9))

system.m = df.Field(mesh, dim=3, norm=4e5,
value=lambda p: (-1e9*p[1], 1e9*p[0], 0.1))

In [4]: md = mc.MinDriver()
md.drive(system)

Running OOMMF (ExeOOMMFRunner) [2022/02/07 10:00]... (0.5 s)

In [5]: system.m.orientation.plane('z').mpl()

In [6]: system.energy += mm.Zeeman(H=(1e4, 0, 0))
md.drive(system)
system.energy.zeeman.H = (0, 0, 0)
td = mc.TimeDriver()
td.drive(system, t=5e-9, n=500)

Running OOMMF (ExeOOMMFRunner) [2022/02/07 10:00]... (0.4 s)
Running OOMMF (ExeOOMMFRunner) [2022/02/07 10:00]... (5.1 s)

In [7]: system.table.data[['t', 'mx', 'my', 'mz', 'E']]
Out [7]:
```

	t	mx	my	mz	E
0	1.00e+11	2.22e-01	4.80e-03	2.16e-01	1.52e+17
1	2.00e+11	2.17e-01	1.02e-02	2.14e-01	1.52e+17
..
498	4.99e+09	-2.39e-03	1.12e-03	2.03e-01	1.49e+17
499	5.00e+09	-2.42e-03	1.21e-03	2.03e-01	1.49e+17

500 rows x 5 columns