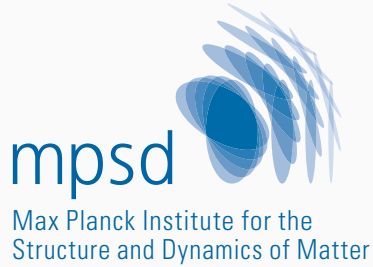
Research software engineering (Ubermag)



Hans Fangohr

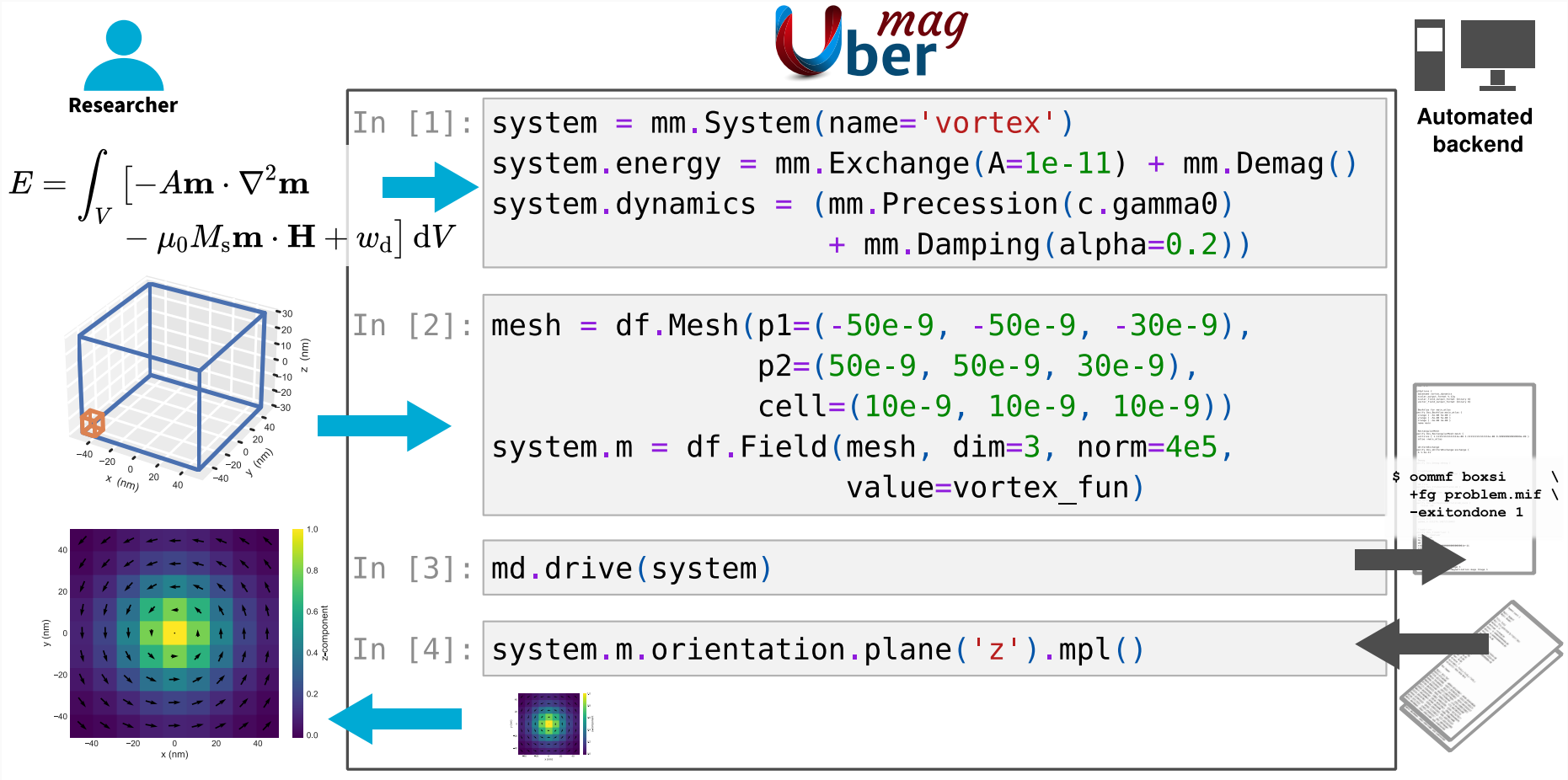
SSU Computational Science
Max Planck Institute for the Structure and Dynamics of Matter
Hamburg, Germany
and
University of Southampton, UK
2022-05-12

hans.fangohr@mpsd.mpg.de
@ProfCompMod

Outline

Introduction Ubermag

Ubermag



- 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)

Physics

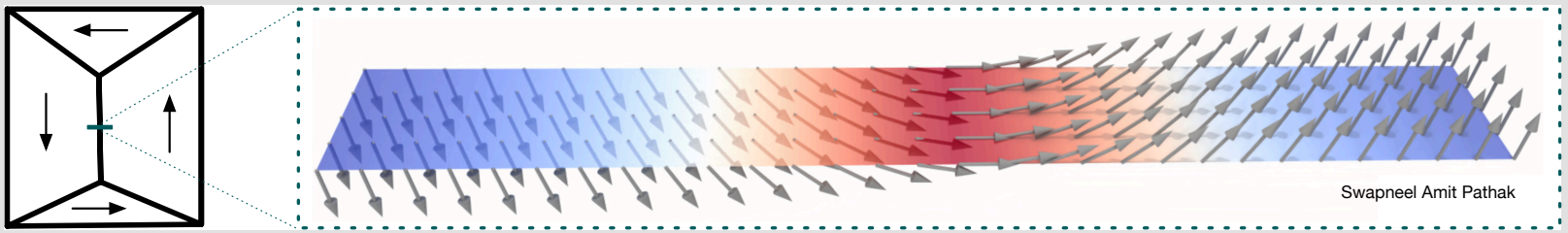
- magnetism at the nanometre to micrometre length scale
- physical property of interest: magnetisation vector field $\mathbf{m}$

$$\mathbf{m} : \mathbb{R}^3 \to \mathbb{R}^3$$

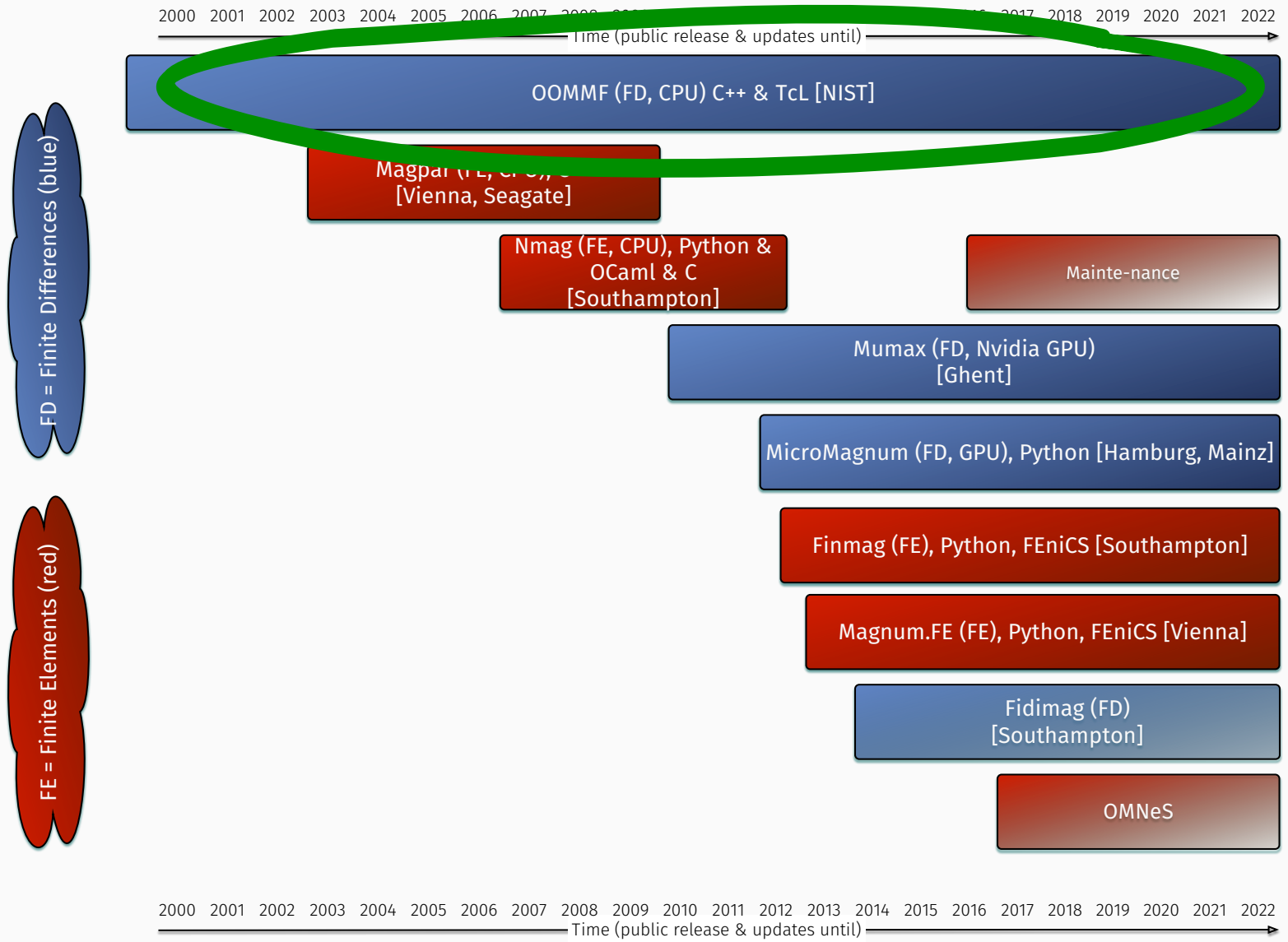
$$\mathbf{r} \mapsto \mathbf{m}(\mathbf{r})$$

Example:

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



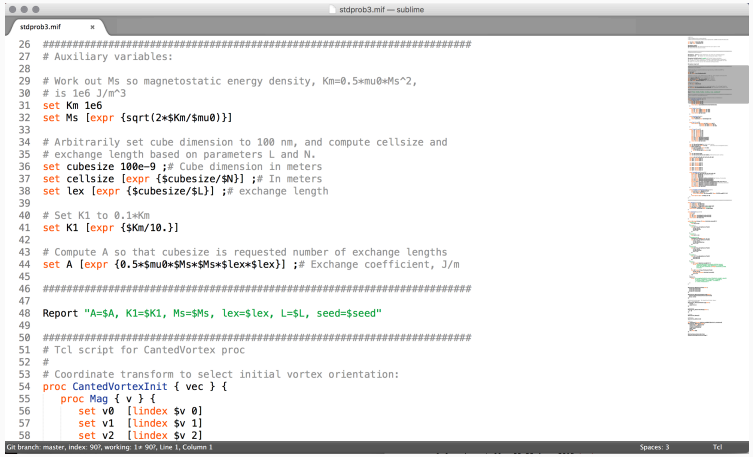
Existing micromagnetic simulation packages



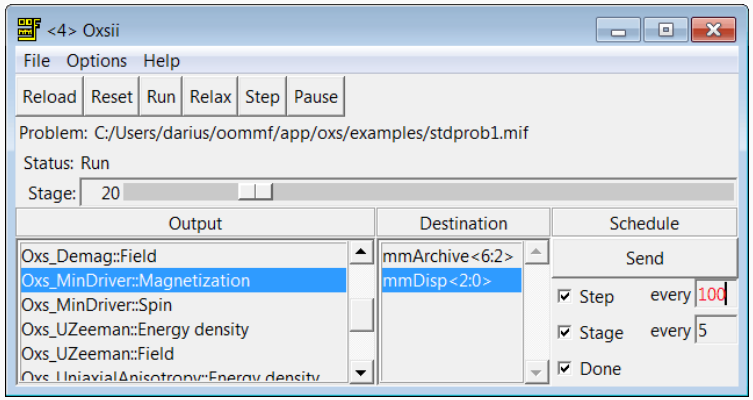
OOMMF: Object Oriented Micromagnetic framework

- 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/`

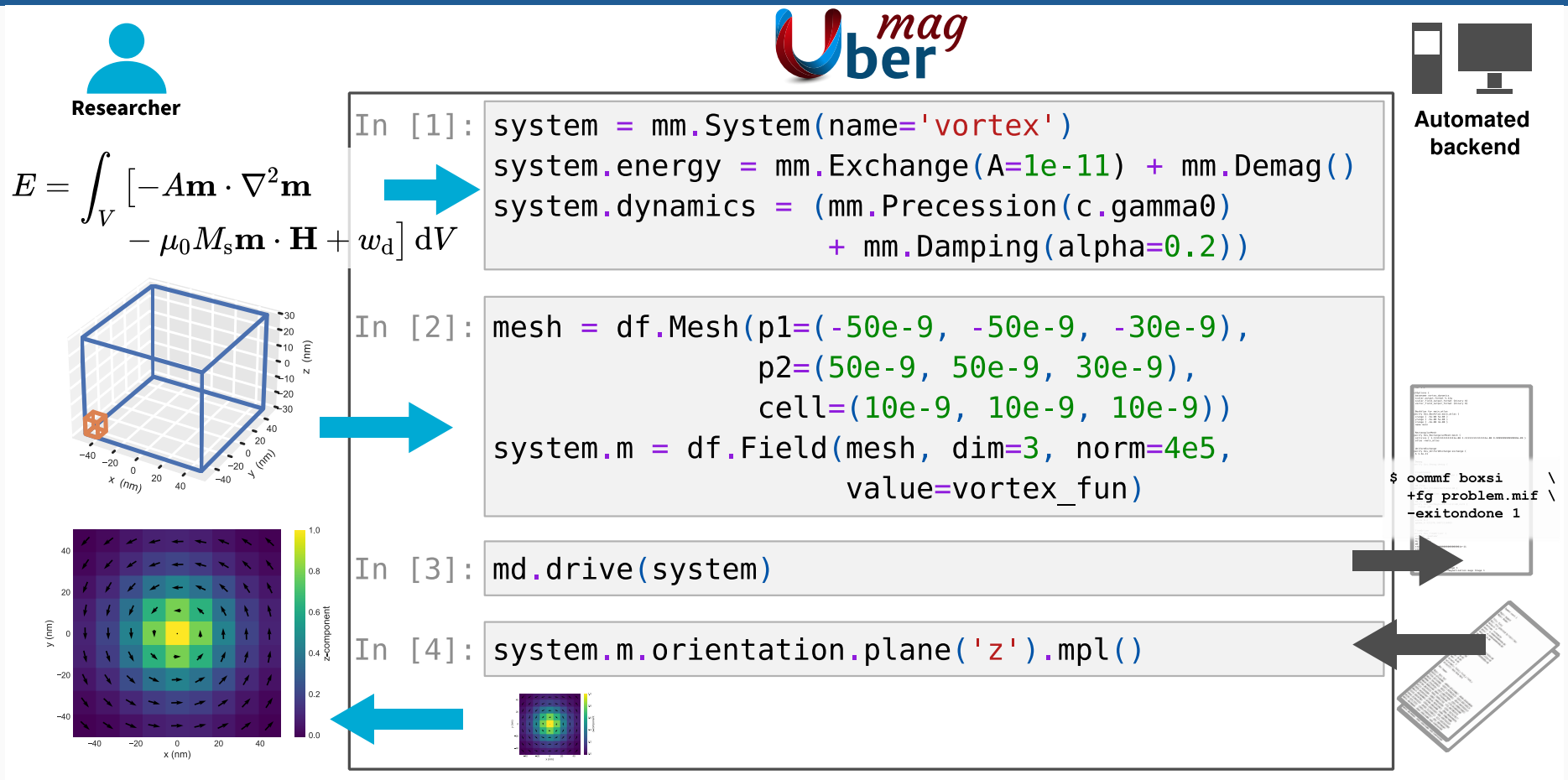
Problem specification (Tcl):



GUI for interactive control and data analysis (Tk):



Ubermag is interface to OOMMF



Ubermag team:

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

Motivation for Ubermag

Ubermag to improve effectiveness of research procedure

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

Beg et.al., *Ubermag: Toward More Effective Micromagnetic Workflows*, 10.1109/TMAG.2021.3078896 (2022)

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

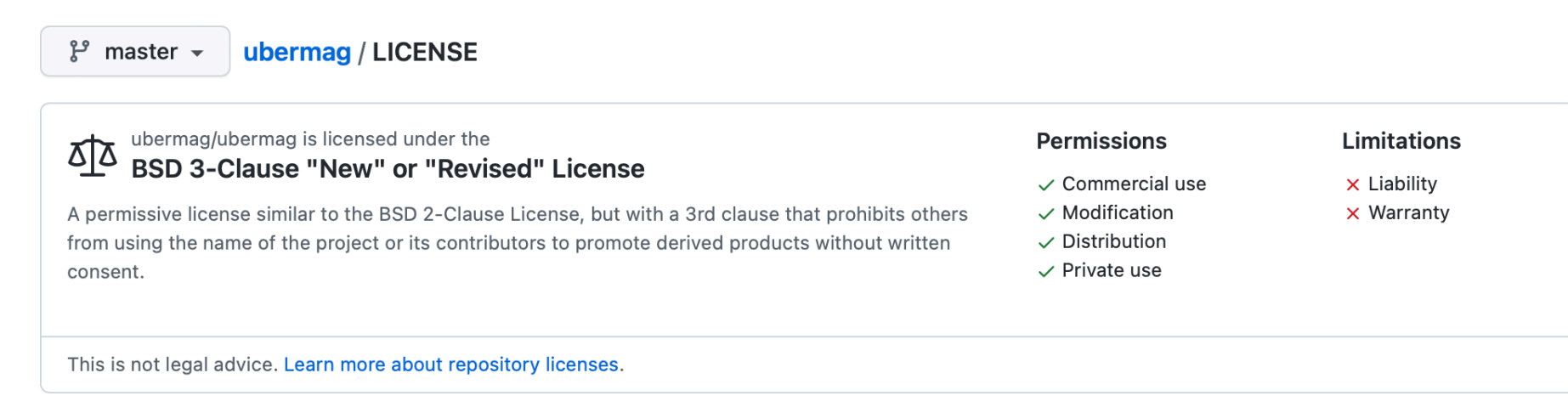
- Keep input data in (git) repository
  (such as notebooks, additional data files & scripts)

Beg et.al., *Using Jupyter for reproducible scientific workflows*, 10.1109/MCSE.2021.3052101 (2021)

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)



Testing: have automated tests

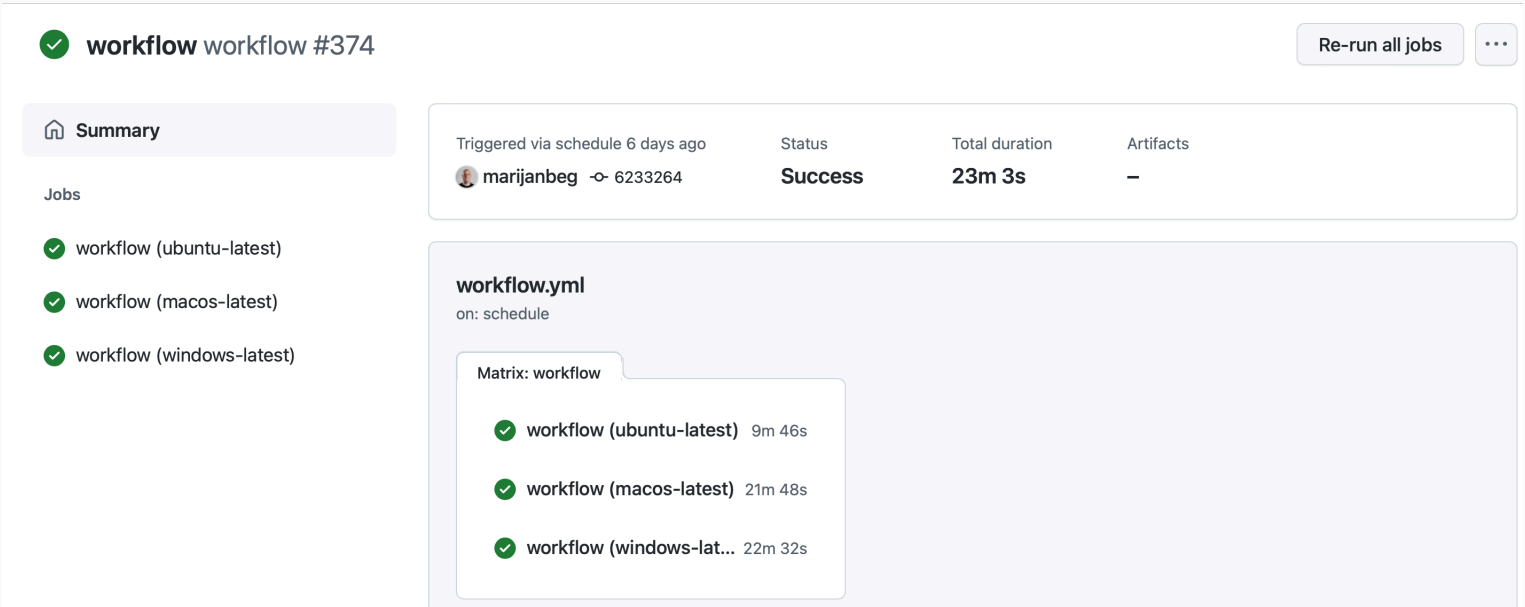
- Idea: in addition to production code, write test functions that test the production code.
- Unit tests, system tests, regression tests
- use `py.test` to collect and execute tests:

```
...
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



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.

Pre-commit hooks

- 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

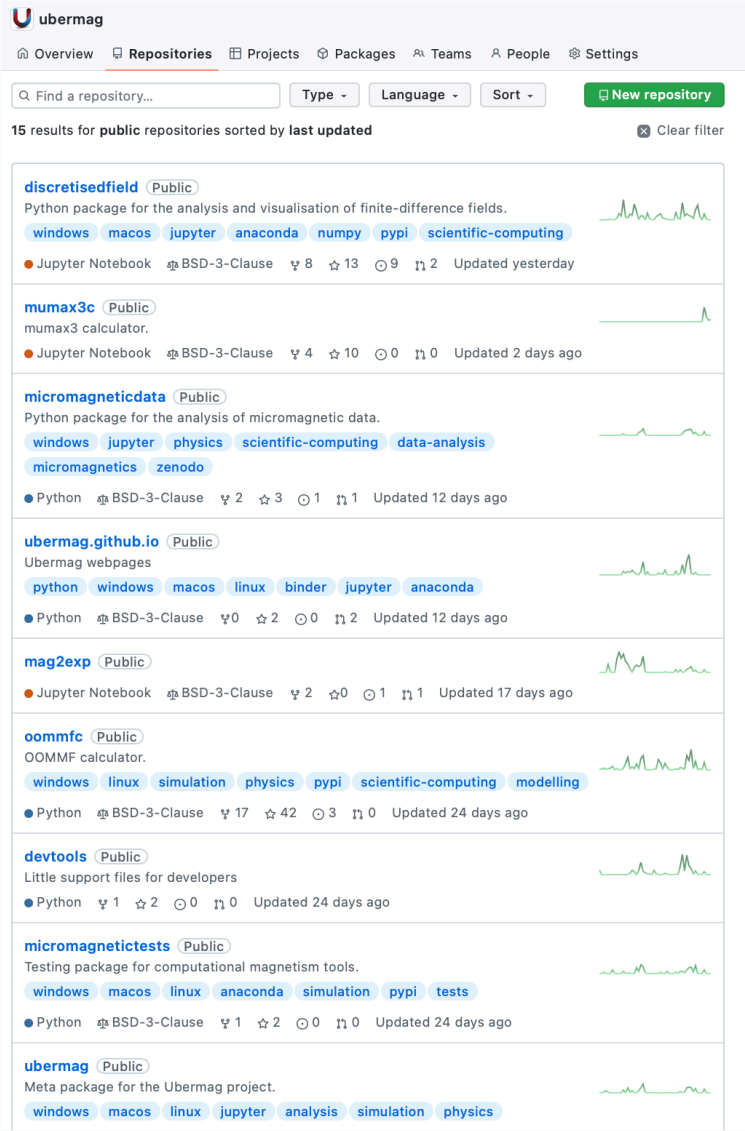
Split project into multiple re-usable python packages

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)



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

Interface design

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*

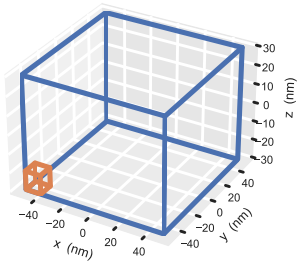
- given material parameters, geometry and discretisation
- solve numeric version of physics problem through existing simulation packages
  - OOMMF (`oommfc`) / mumax3 (`mumax3c`) / ...?

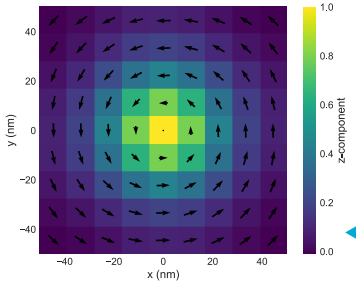
Beg, Pepper, Fangohr: *User interfaces for computational science: a domain specific language for OOMMF embedded in Python*, 10.1063/1.4977225, (2017)

Ubermag overview

Researcher

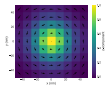
$$E = \int_V \left[-A\mathbf{m} \cdot \nabla^2 \mathbf{m} - \mu_0 M_\mathrm{s} \mathbf{m} \cdot \mathbf{H} + w_\mathrm{d} \right] \mathrm{d}V$$



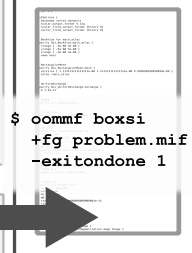


Ubermag

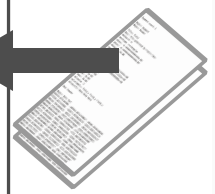
```
In [1]: system = mm.System(name='vortex')
        system.energy = mm.Exchange(A=1e-11) + mm.Demag()
        system.dynamics = (mm.Precession(c.gamma0)
                           + mm.Damping(alpha=0.2))

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



```
$ oommf boxsi      \
  +fg problem.mif \
  -exitondone 1
```



Interface design 2

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

Documentation: Jupyter notebooks

ubermag.github.io/examples/notebooks/09-tutorial-standard-problem5.html

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

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

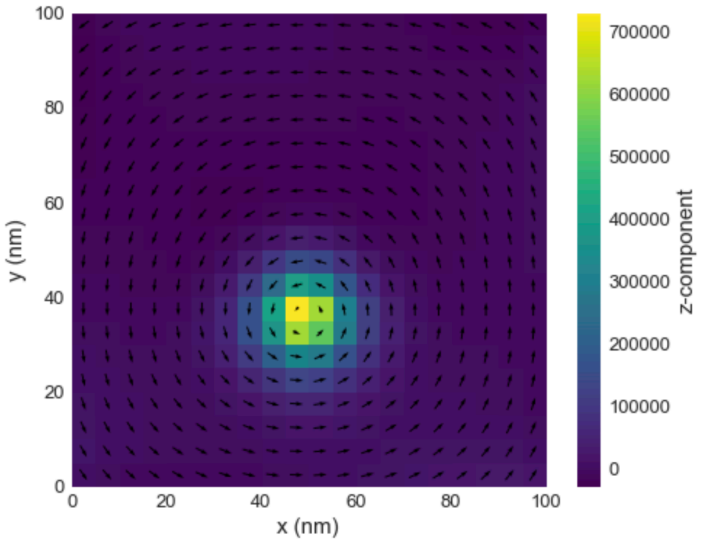
Now, we can drive the system for $8\,\text{ns}$ and save the magnetisation in $n = 100$ steps.

[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 $8\,\text{ns}$ is now displaced from the centre.

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



Documentation: Jupyter notebooks

- 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`)

Fangohr et.al., *Data exploration and analysis with Jupyter*, 10.18429/JACoW-ICALEPCS2019-TUCPR02 (2020)

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) ==========================================
```


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

Testing the documentation (Notebooks with NBVAL)

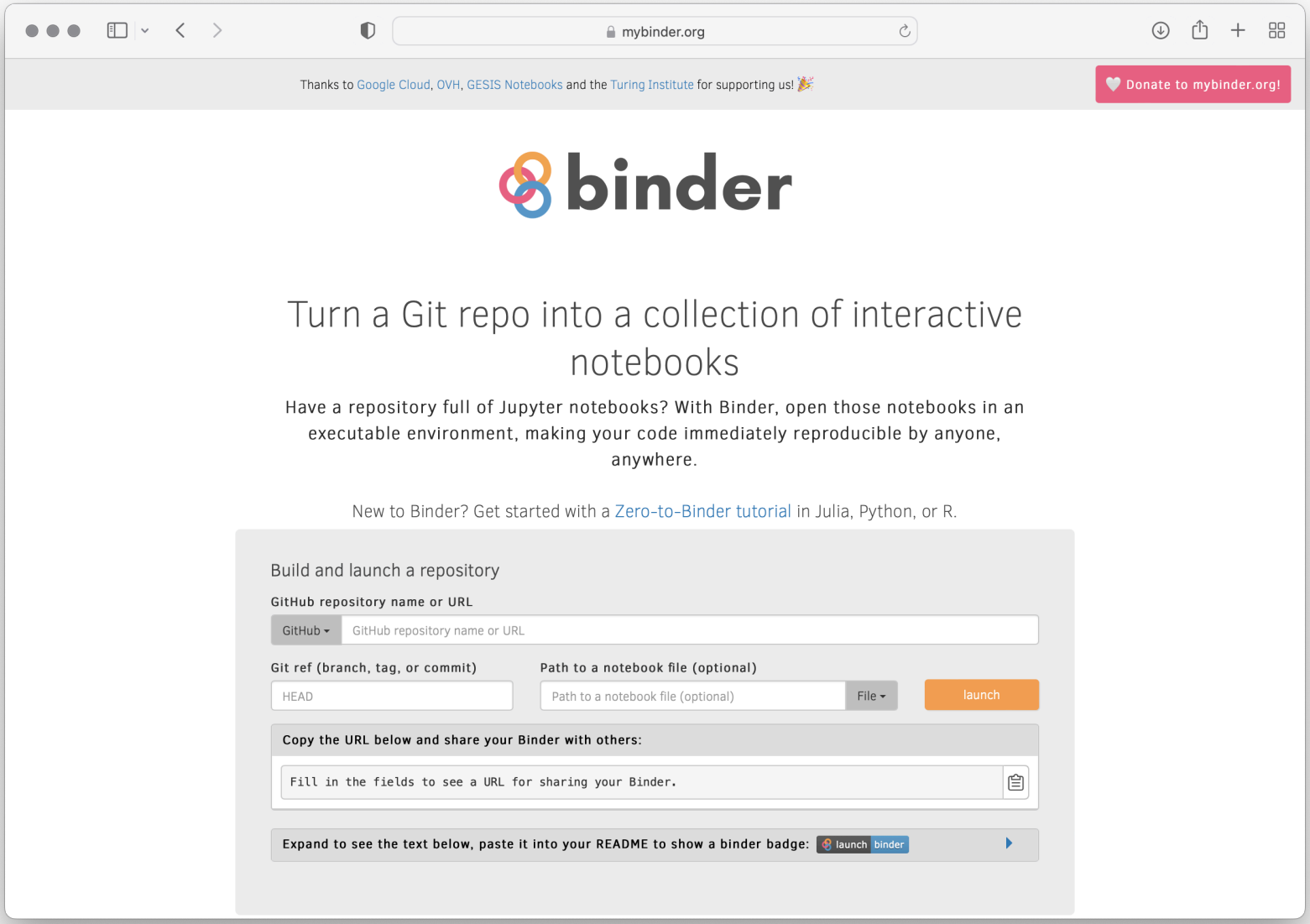
NBVAL to *VALidate* NoteBooks

- 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



Interactive documentation & tutorials (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`

Research Software Engineering Ubermag: relevant publications

- 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