Hans Fangohr[1,2] (@ProfCompMod)
Marijan Beg[1,2], Sergii Mamedov[1], Ryan A. Pepper[2], David Cortés-Ortuño[2], Thomas Kluyver[1,2]

DRIVING SIMULATION AND DATA ANALYSIS OF MAGNETIC NANOSTRUCTURES THROUGH JUPYTER NOTEBOOK

[1]European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, Germany
[2]University of Southampton, Highfield, SO17 1BJ Southampton, UK

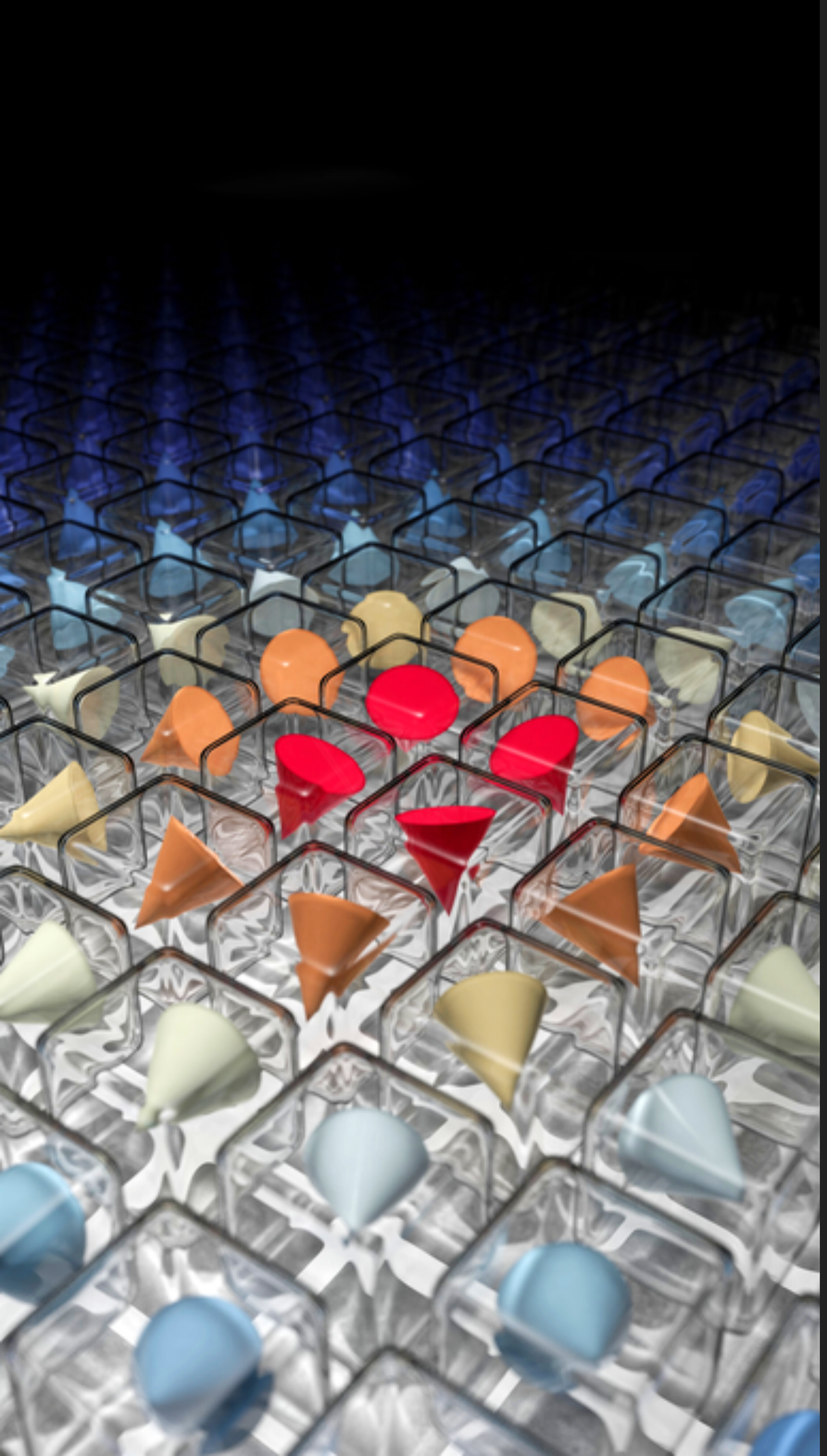
25 October 2018, PyConDE & PyData, Karlsruhe, Germany

Hans Fangohr[1,2] (@ProfCompMod)
Marijan Beg[1,2], Sergii Mamedov[1], Ryan A. Pepper[2], David Cortés-Ortuño[2], Thomas Kluyver[1,2]

REPRODUCIBLE COMPUTATIONAL SCIENCE WITH JUPYTER

[1]European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, Germany
[2]University of Southampton, Highfield, SO17 1BJ Southampton, UK

25 October 2018, PyConDE & PyData, Karlsruhe, Germany

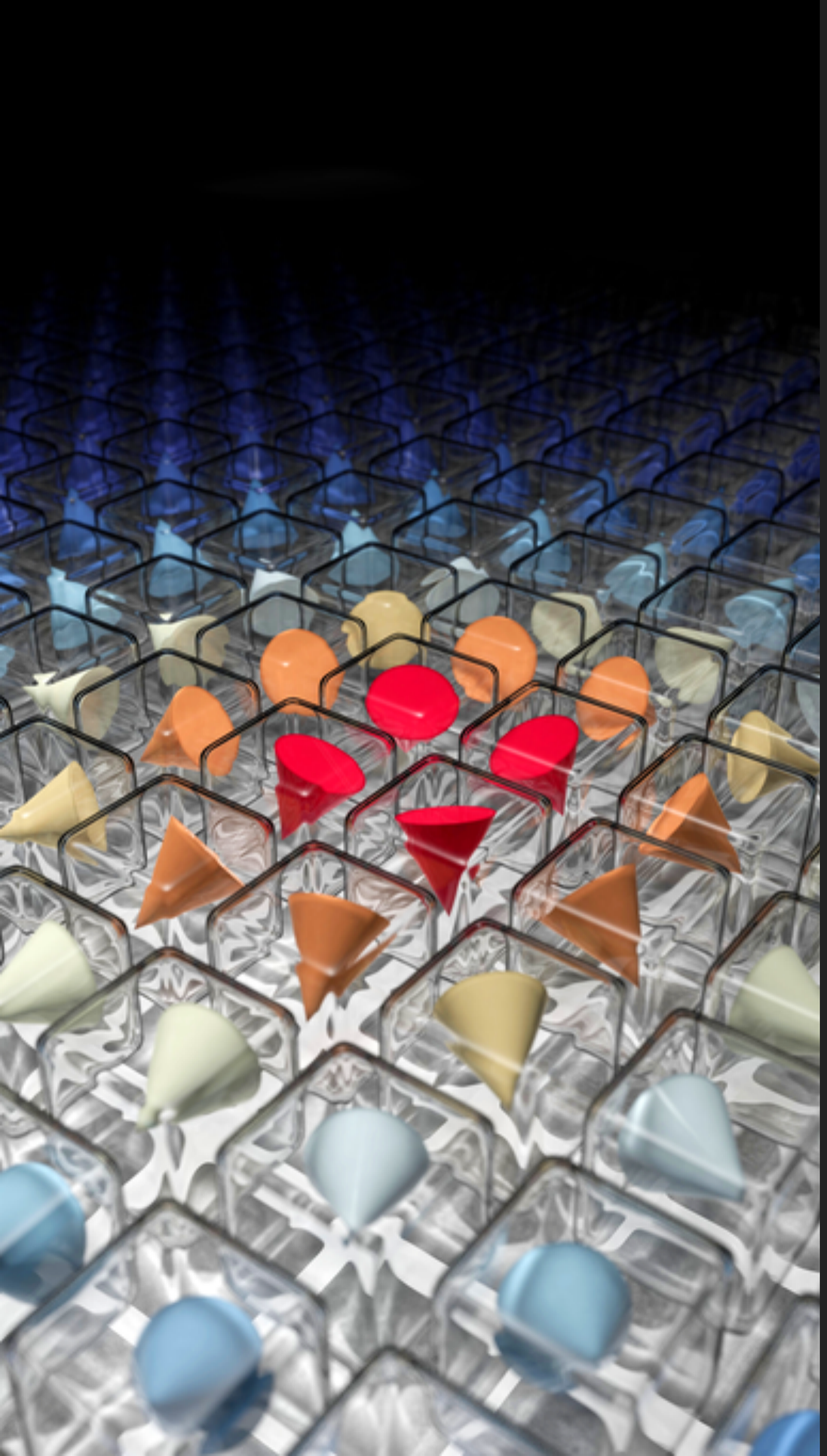


PART 0

OUTLINE

OUTLINE

1. Motivation
2. Basics of computational micromagnetics, the Object-Oriented MicroMagnetic Framework (OOMMF)
3. Typical computational workflow
4. Python interface
5. Demo
6. Discussion and Summary



PART 1

MOTIVATION

MAKE COMPUTATIONAL SCIENCE MORE EFFECTIVE

- Computational science and data science of increasing importance
- Looking for methods to make it more effective/better:
- Faster to use, more accessible, more reproducible

For upcoming reproduce-along-session in 10 minutes, open browser at
https://tinyurl.com/yapnotyb

REPRODUCIBILITY IN COMPUTATIONAL SCIENCE & DATA SCIENCE

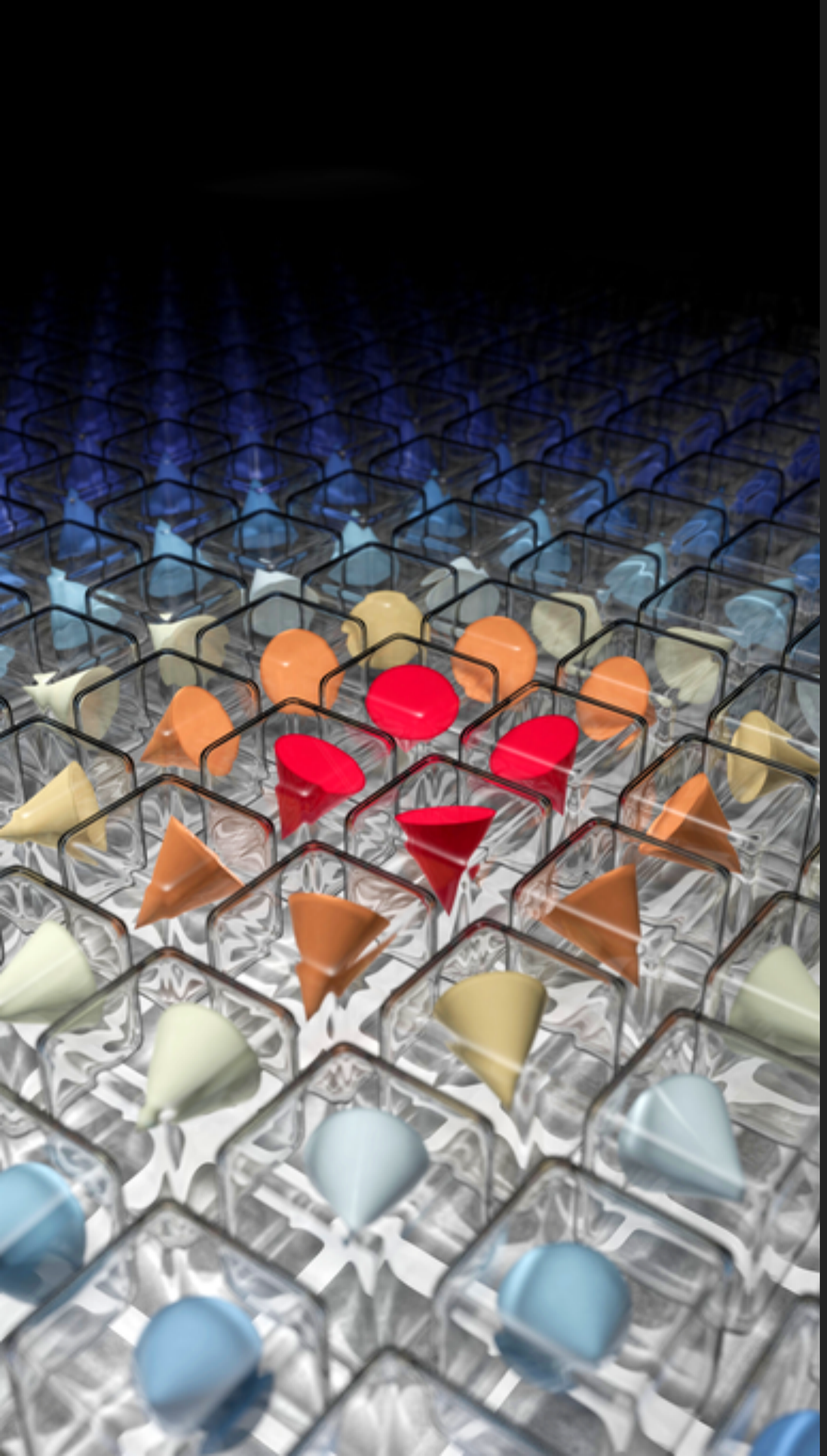
- Often results are not reproducible. Why not?
  - Complex workflow: simulation codes, configuration files, execution of many runs, post-processing, image creation, report/publication writing
  - "Sharing data and software gives away competitive edge."
  - "Don't know how to."
  - "It is too much work."

For upcoming reproduce-along-session in 10 minutes, open browser at
https://tinyurl.com/yapnotyb

NEXT GENERATION EXECUTABLE DOCUMENTS: BINDER

- Execution of (somebody else's) documents in browser
- Realised by Binder (tailored JupyterHub, can run in Cloud).

For upcoming reproduce-along-session in 10 minutes, open browser at
https://tinyurl.com/yapnotyb

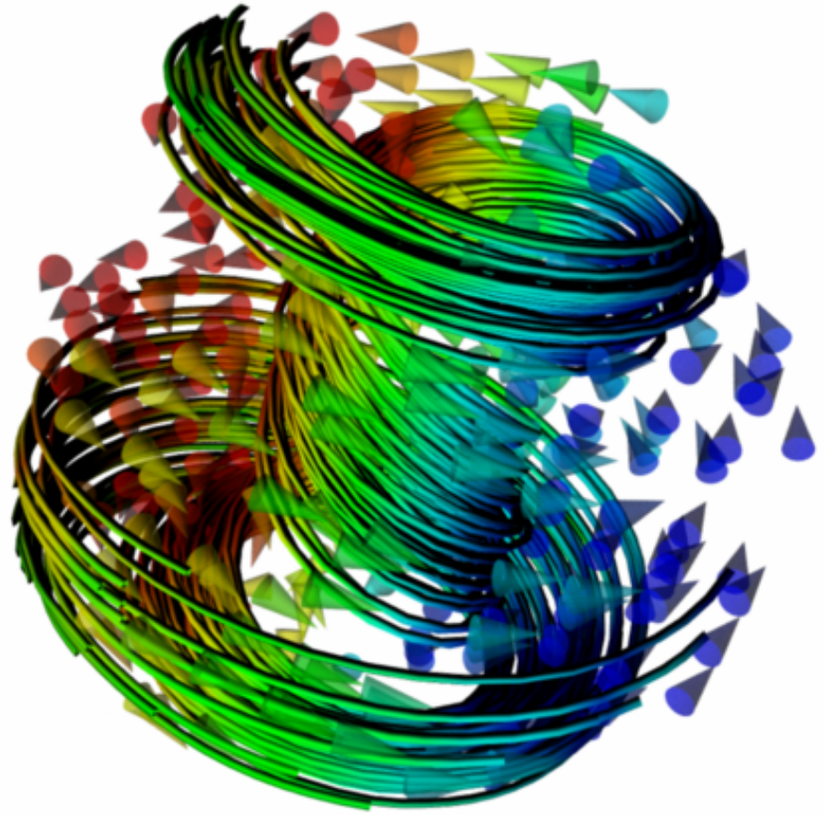


PART 2

MICROMAGNETICS AND OOMMF

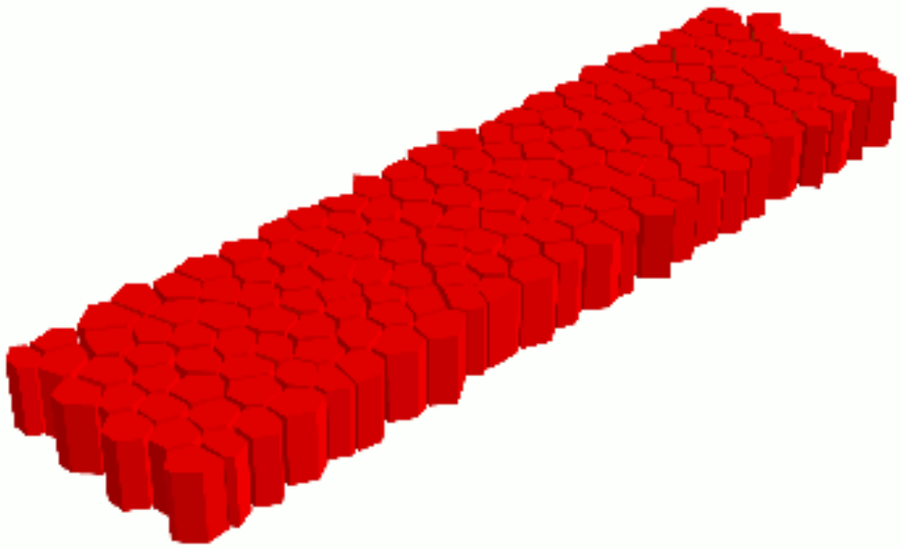
MICROMAGNETICS

"… is the study, modelling, and simulation of magnetic materials and their behaviour at the nanometer scale."

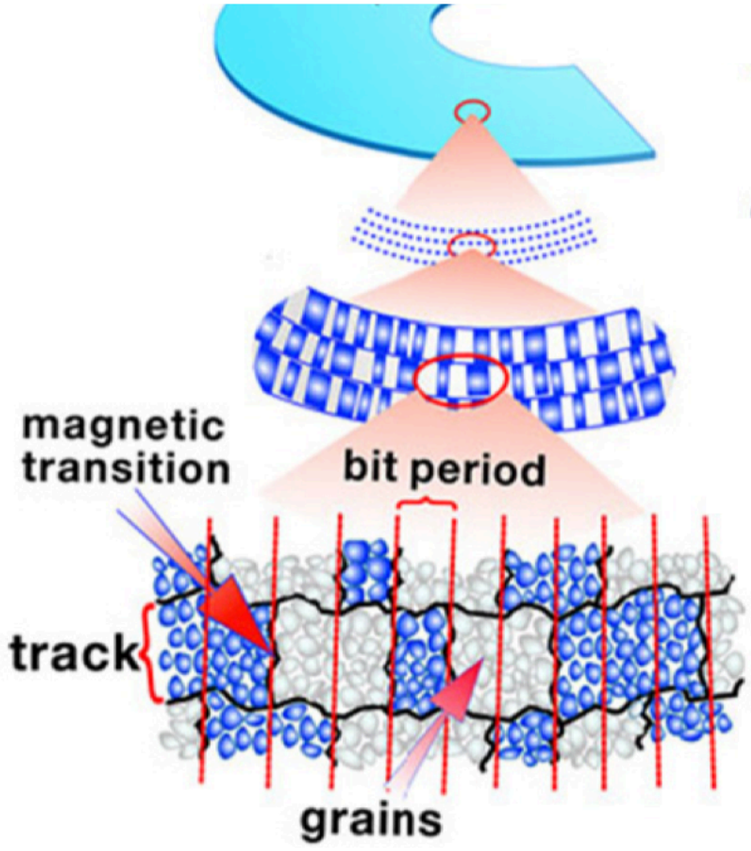


STUDY OF MAGNETIC NANOSTRUCTURES

- Interesting complex system with tuneable parameters and experiments
- Real applications, including
  - magnetic data storage (hard disk, tape)
  - cancer therapy
  - low energy magnetic logic (spintronics)

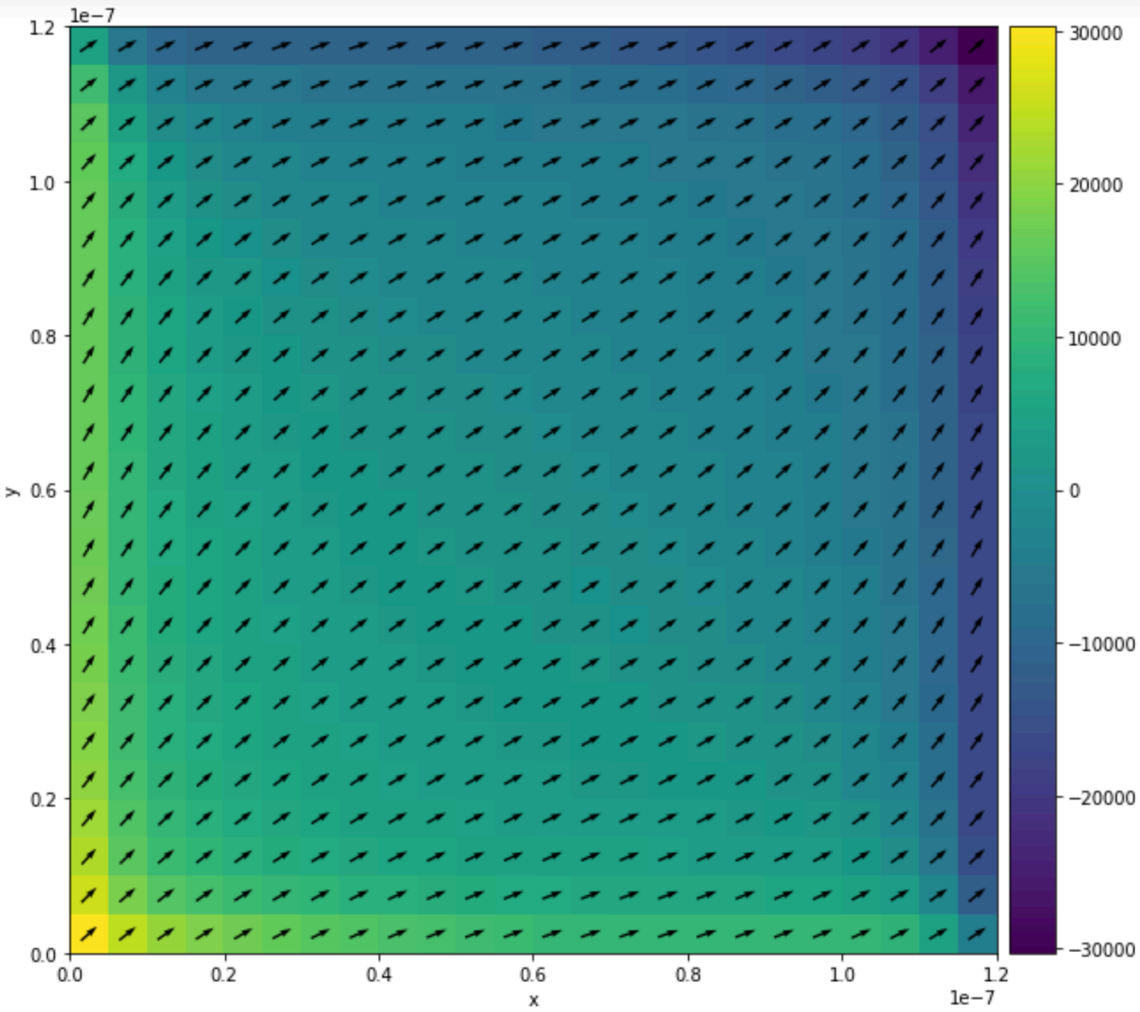


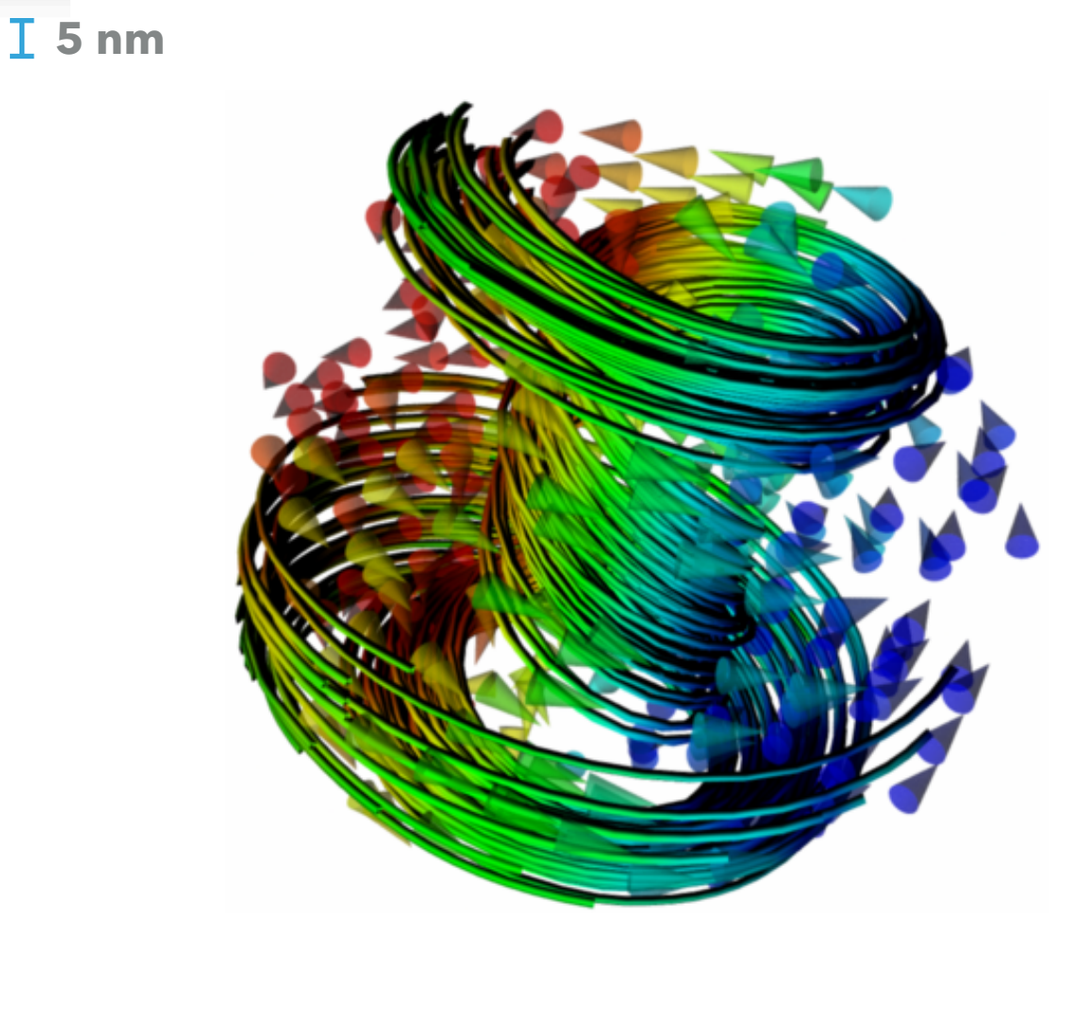
Curtis & Fangohr (2011)



Griffiths et al, J. Phys. D: Appl. Phys. 46 (2013) 503001

MICROMAGNETIC SIMULATION RESULTS





MAGNETISATION

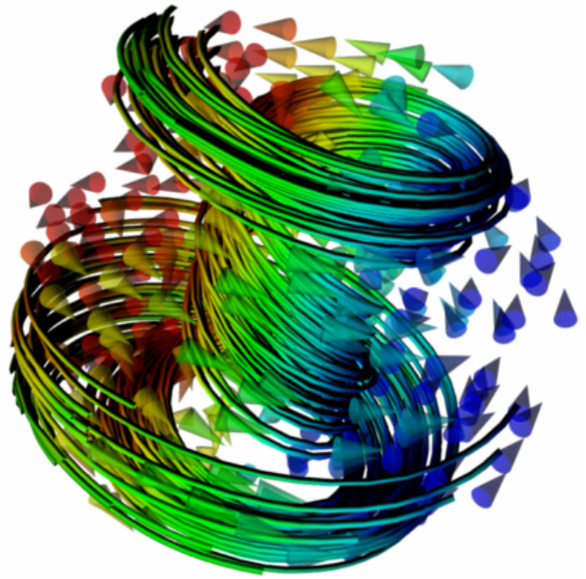
- Continuous vector field

$$\mathbf{M} = \mathbf{M}(\mathbf{r}, t) \qquad \mathbf{M} : \mathbb{R}^3 \to \mathbb{R}^3$$

- Normalised magnetisation

$$\mathbf{m}(\mathbf{r}) = \frac{\mathbf{M}(\mathbf{r})}{M_\mathrm{s}}$$

- $M_\mathrm{s} = |\mathbf{M}|$ saturation magnetisation



- In general, magnetisation is a function of both space and time.

HAMILTONIAN – THE PHYSICS

- Energy density

$$w(\mathbf{m}) = w_1(\mathbf{m}) + w_2(\mathbf{m}) + w_3(\mathbf{m}) + \cdots = \sum_i w_i(\mathbf{m})$$

- Total energy through integration of energy density

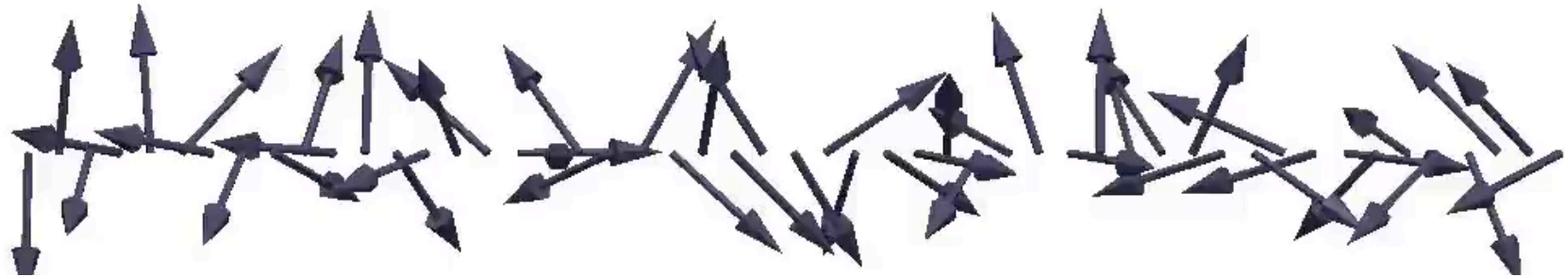
$$E = \int_V w(\mathbf{m}) dV$$

- Different physical effects originate from different energy terms (examples on next slide)

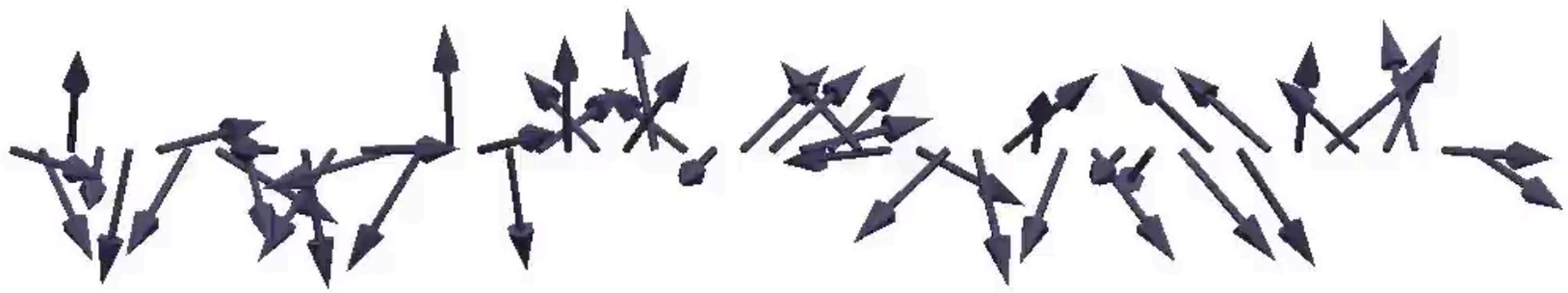
MICROMAGNETIC EXAMPLES (1D TOY MODEL)

- Applied field (up)

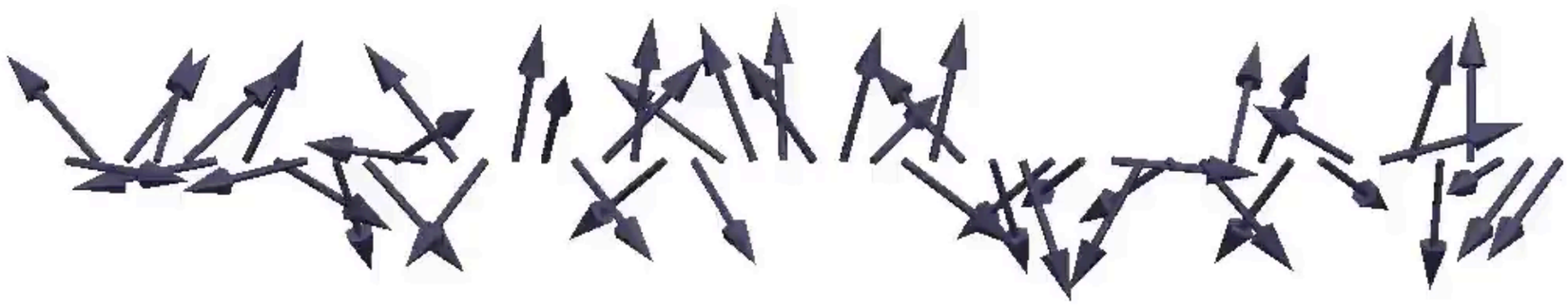
- Parallel alignment



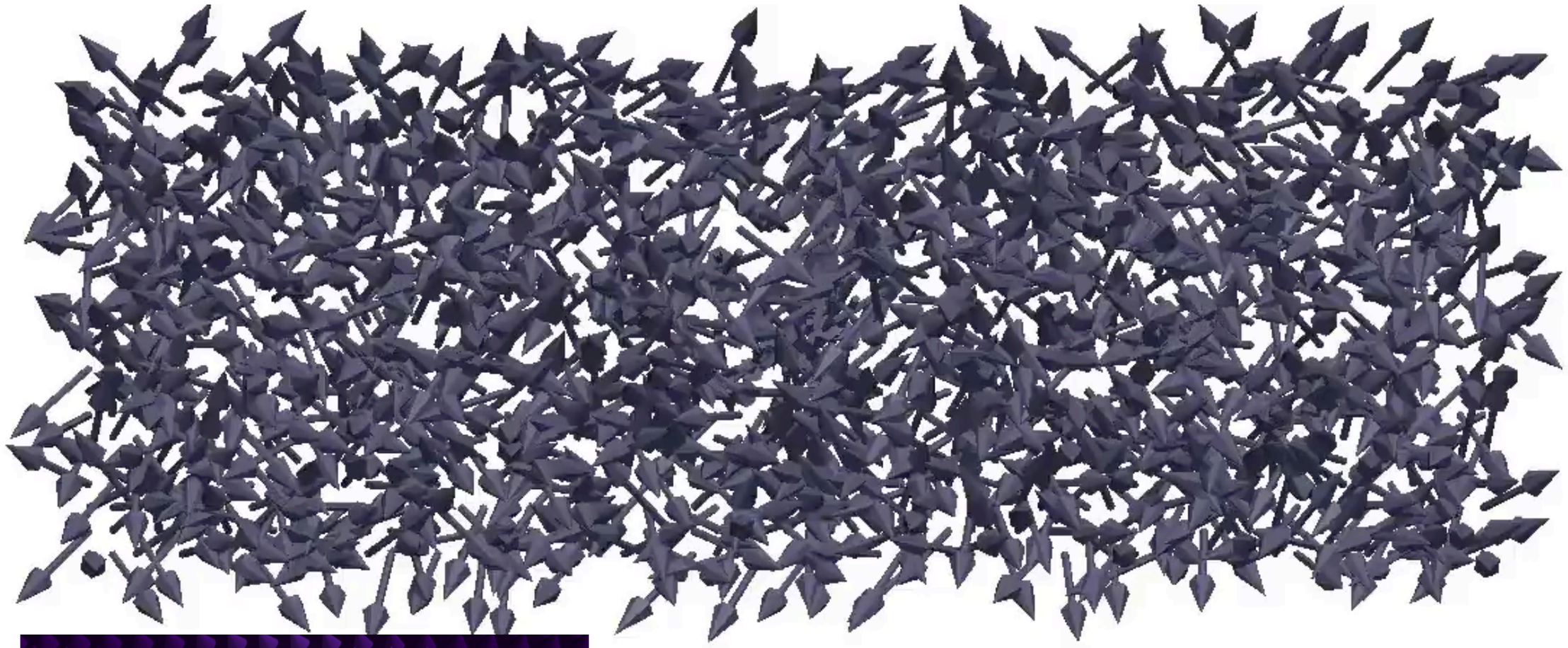
- Orthogonal alignment

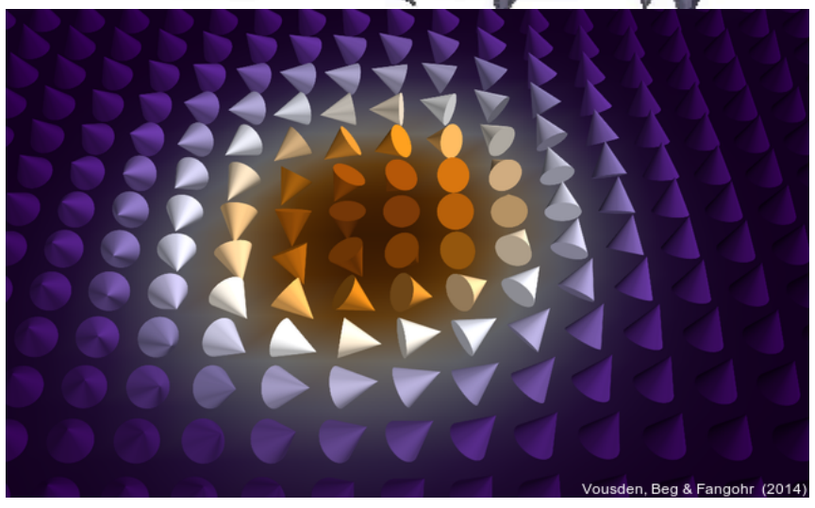


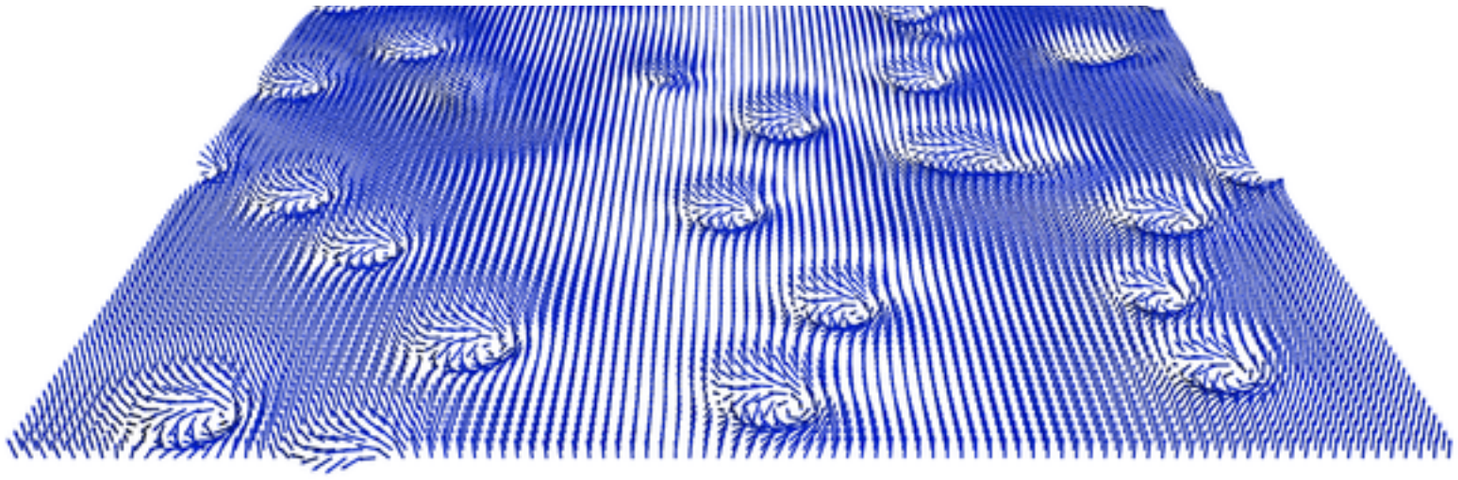
- Par. & Orth. alignment



MORE COMPLICATED 2D SAMPLE – SKYRMION







Marijan Beg *et al.* Nature's Scientific Reports 7, 17137 (2015)

Vousden, Beg, & Fangohr (2014)

EQUATIONS FOR MICROMAGNETICS (non-linear partial integro-differential equation)

$$\phi(\mathbf{r}') = \frac{M_\mathrm{s}}{4\pi}\left(\underbrace{-\int_\Omega \frac{\nabla\cdot\mathbf{m}(\mathbf{r}')}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}V}_{\text{volume term}} + \overbrace{\int_{\partial\Omega}\frac{\mathbf{m}(\mathbf{r}')\cdot\mathbf{n}}{||\mathbf{r}-\mathbf{r}'||}\,\mathrm{d}S}^{\text{surface term}}\right)$$

$$\mathbf{H}_\mathrm{d}(\mathbf{r}) = -\nabla\phi(\mathbf{r})$$

- Effective field

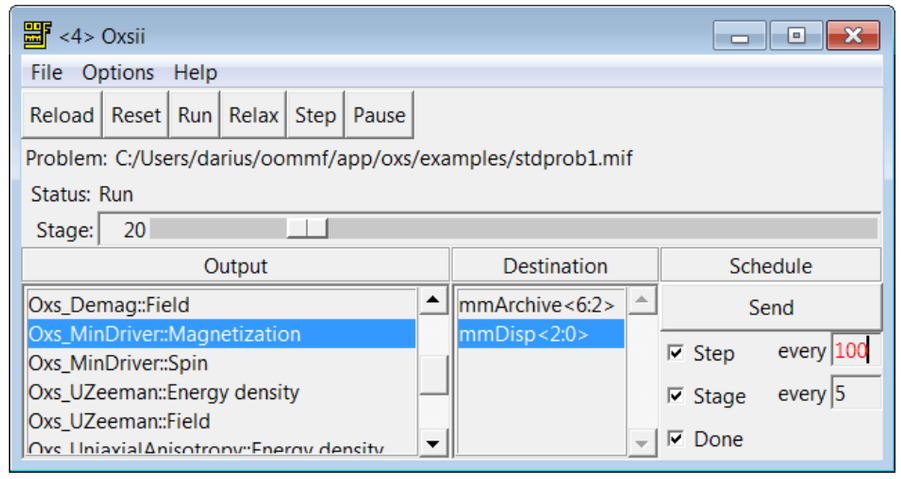
$$H_\mathrm{eff}(\mathbf{m}) = \underbrace{\frac{2A}{\mu_0 M_\mathrm{s}}\nabla^2\mathbf{m}}_{\text{exchange}} - \overbrace{\frac{2D}{\mu_0 M_\mathrm{s}}(\nabla\times\mathbf{m})}^{\text{DMI}} + \underbrace{\mathbf{H}}_{\text{Zeeman}} + \overbrace{\mathbf{H}_\mathrm{d}}^{\text{demagnetisation}}$$

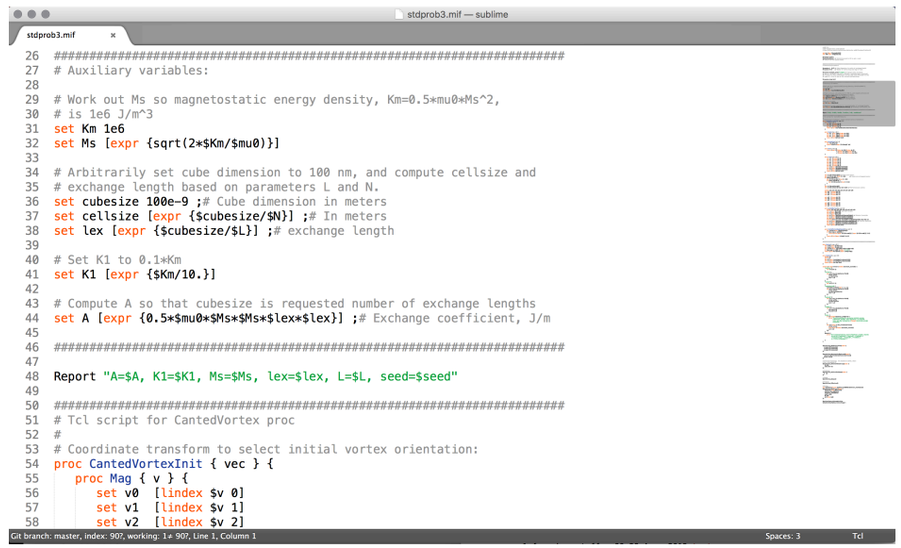
- LLG

$$\frac{\partial\mathbf{m}}{\partial t} = \underbrace{\gamma^*\mathbf{m}\times\mathbf{H}_\mathrm{eff}}_{\text{precession}} + \overbrace{\alpha\mathbf{m}\times\frac{\partial\mathbf{m}}{\partial t}}^{\text{damping}} + \underbrace{u(|\mathbf{m}|-1)\frac{\mathbf{m}}{|\mathbf{m}|}}_{\text{norm correction}}$$

OOMMF (OBJECT ORIENTED MICROMAGNETIC FRAMEWORK)

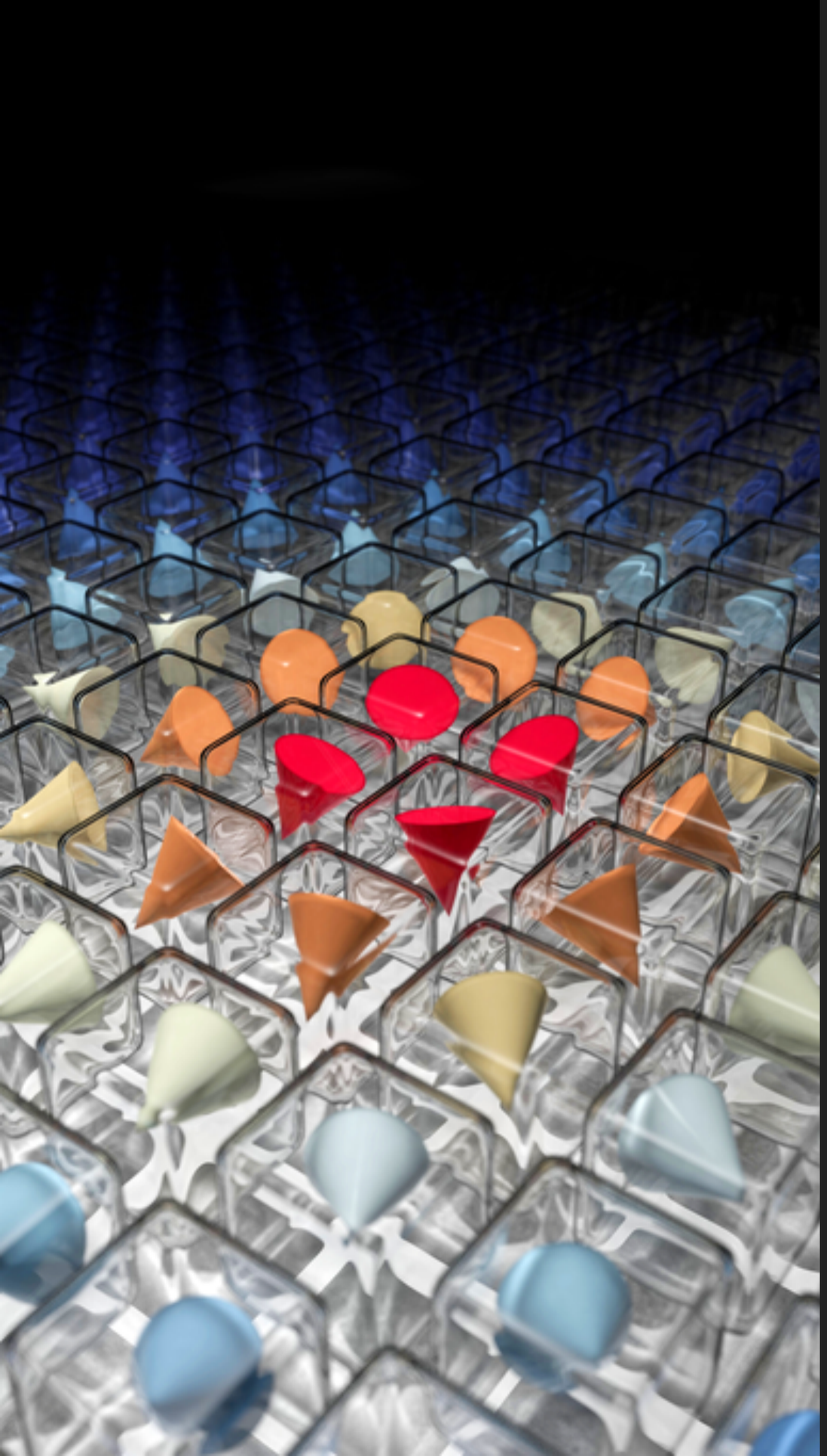
- Probably the most widely used micromagnetic simulation tool
- Developed at National Institute for Standards and Technology (NIST), US, since ~1998 by Michael Donahue & Don Porter
- Cited over 2200 times in scientific publications
- Written in C++ & Tcl
- math.nist.gov/oommf/





EXAMPLE OOMMF CONFIGURATION FILE

```
# MIF 2.1

# BoxAtlas
Specify Oxs_BoxAtlas:atlas {
  xrange {0 1e-07}
  yrange {0 1e-07}
  zrange {0 1e-07}
  name atlas
}

# RectangularMesh
Specify Oxs_RectangularMesh:mesh {
  cellsize {6.25e-09 6.25e-09 6.25e-09}
  atlas Oxs_BoxAtlas:atlas
}

# UniformExchange
Specify Oxs_UniformExchange {
  A 1.4046639231824416e-10
}

# UniaxialAnisotropy
Specify Oxs_UniaxialAnisotropy {
  K1 100000.0
  axis {0 0 1}
}

# Demag
Specify Oxs_Demag {}

# m0 file
Specify Oxs_FileVectorField:m0file {
   atlas :atlas
   file m0.omf
}

# CGEvolver
Specify Oxs_CGEvolve {}

# MinDriver
Specify Oxs_MinDriver {
  evolver Oxs_CGEvolve
  stopping_mxHxm 0.01
  mesh :mesh
  Ms {
    Oxs_VecMagScalarField {
      field :m0file
    }
  }
  m0 :m0file
  basename stdprob3
  scalar_field_output_format {text %\#.15g}
  vector_field_output_format {text %\#.15g}
}

Destination table mmArchive
Destination mags mmArchive

Schedule DataTable table Stage 1
Schedule Oxs_MinDriver::Magnetization mags Stage 1
```



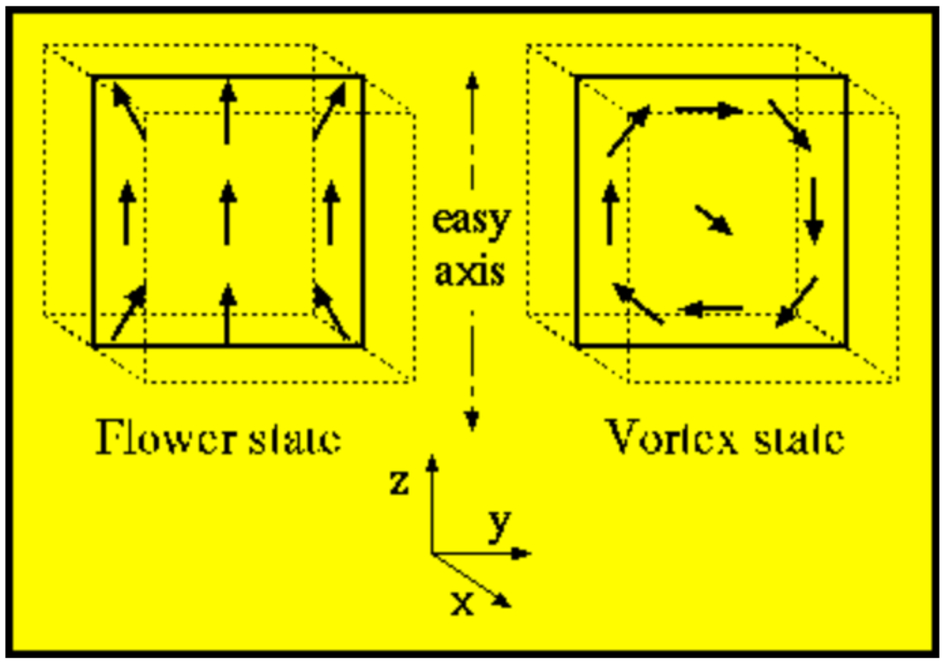
PART 3

TYPICAL COMPUTATIONAL WORKFLOW

STANDARD PROBLEM 3

https://www.ctcms.nist.gov/~rdm/spec3.html

μMAG Standard Problem #3



Comments:

The transition is expected to be found in the neighborhood of $L = 8\ l_{ex}$.

Site Directory

μMAG organization / NIST CTCMS / rmcmichael@nist.gov
20-MAR-1998

Problem proposed by Alex Hubert, University of Erlangen-Nuremberg.

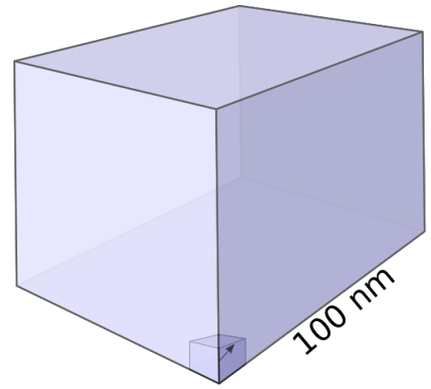
Please send comments to rmcmichael@nist.gov and join the μMAG discussion e-mail list for ongoing discussion.

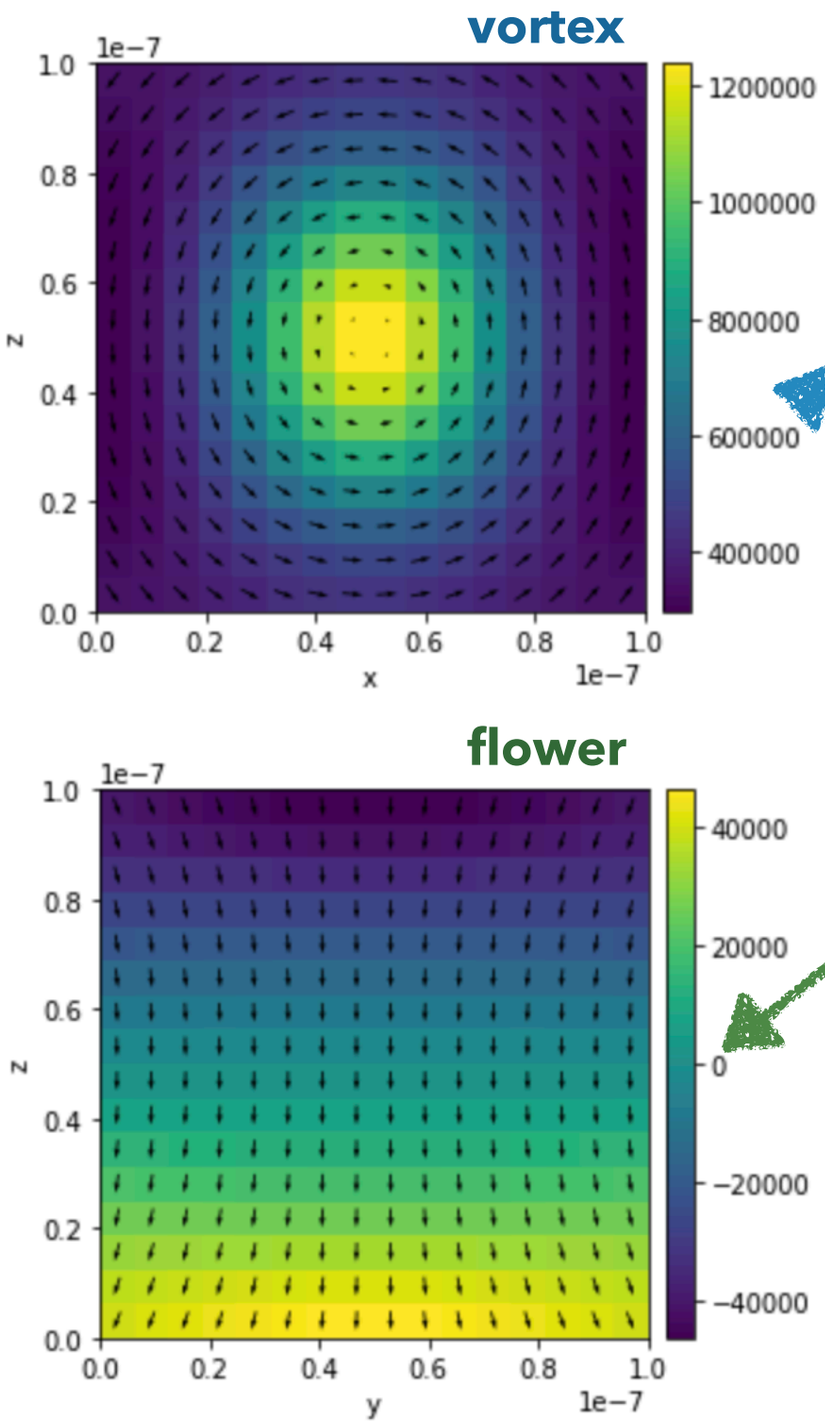
A set of solutions have been submitted.

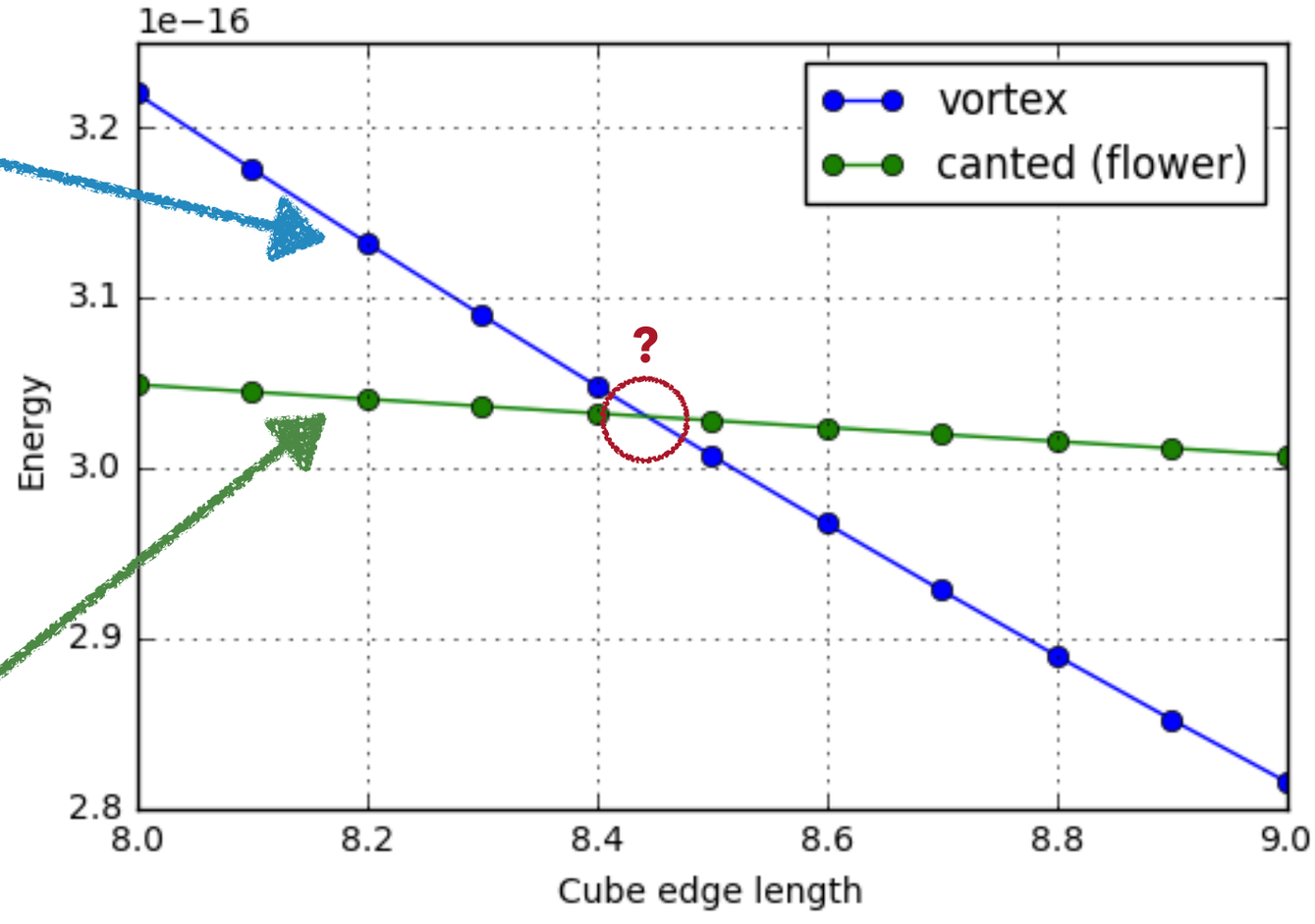
Specifications

This problem is to calculate the single domain limit of a cubic magnetic particle. This is the size L of equal energy for the so-called flower state (which one may also call a splayed state or a modified single-domain state) on the one

RESEARCH QUESTION



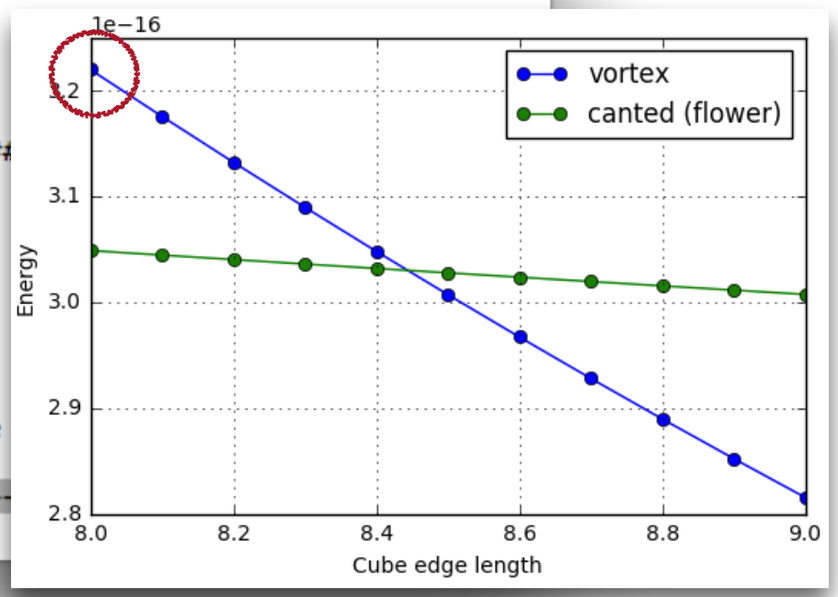




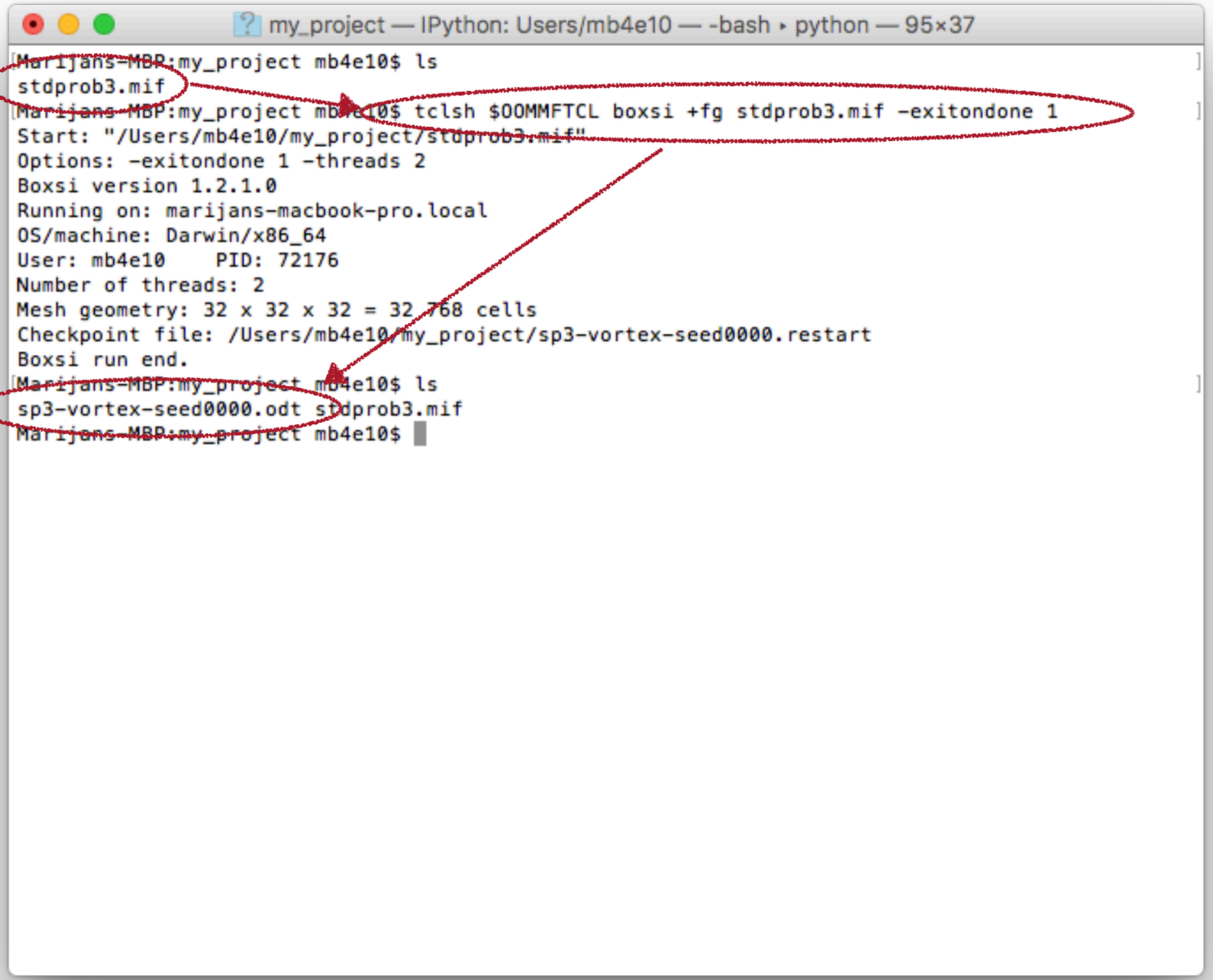
STEP 1: WRITE SIMULATION FILE

my_project — IPython: Users/mb4e10 — emacs -nw stdprob3.mif — 95×37

```
# MIF 2.1
# MIF Example File: stdprob3.mif
# Description: Sample problem description for muMAG Standard Problem #3

set pi [expr {4*atan(1.0)}]
set mu0 [expr {4*$pi*1e-7}]

Parameter seed 0
RandomSeed $seed  ;# Initialize seed to {} to get a seed
## value from the system clock.


########################################################################
# Simulation parameters

Parameter L 8    ;# Cube dimension, in units of exchange length
Parameter N 32    ;# Number of cells along one edge of cube

Parameter initial_state "vortex" ;# Initial state should be
## one of "uniform", "vortex", "canted", "cantedvortex", "twisted",
## "random" or "file <filename>"; in the last case <filename> is the
## name of a file to use as the initial configuration.

Parameter stop 1e-3

########################################################################
# Auxiliary variables:

# Work out Ms so magnetostatic energy density, Km=0.5*mu0*Ms^2,
# is 1e6 J/m^3
set Km 1e6
set Ms [expr {sqrt(2*$Km/$mu0)}]

# Arbitrarily set cube dimension to 100 nm, and compute cellsize
# exchange length based on parameters L and N.
-uu-:---F1  stdprob3.mif   Top L1     (Fundamental)----------
```



STEP 2: RUN SIMULATION



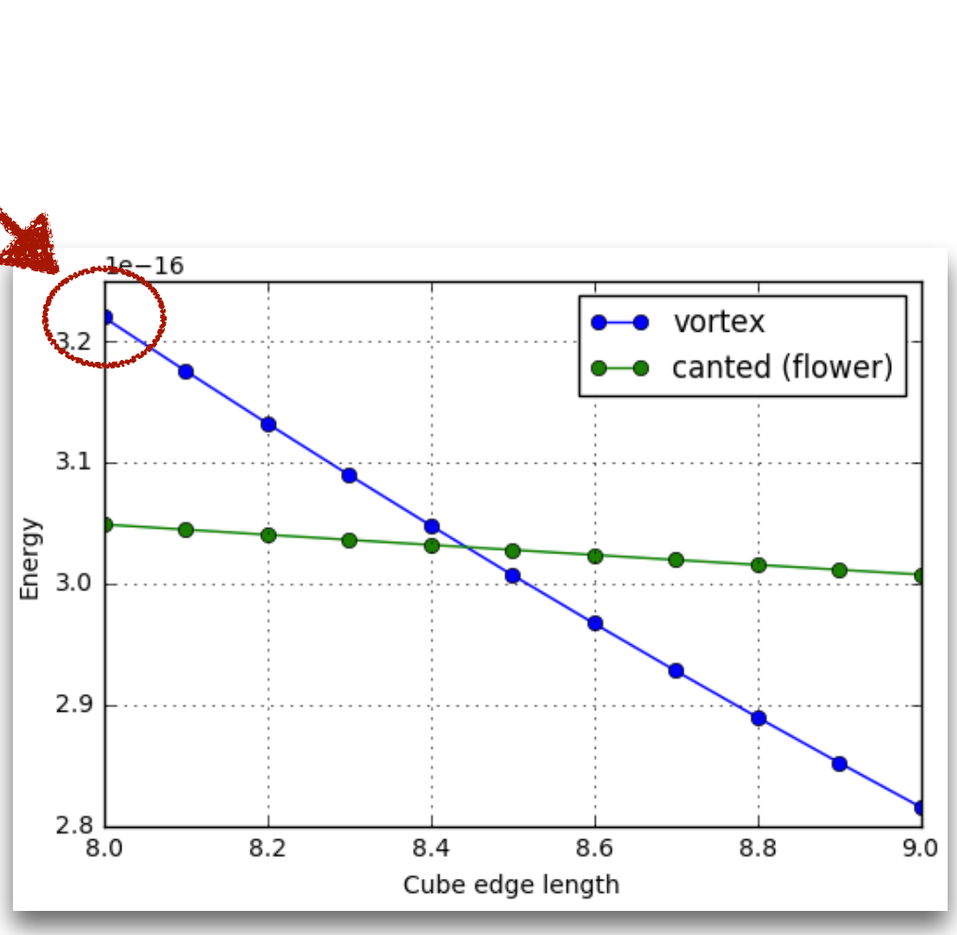
STEP 3: READ DATA

my_project — IPython: Users/mb4e10 — emacs -nw sp3-vortex-seed0000.odt — 95×37

```
# ODT 1.0
# Table Start
# Title: mmArchive Data Table, Wed Nov 16 20:54:28 GMT 2016
# Columns: {Oxs_CGEvolve::Max mxHxm} {Oxs_CGEvolve::Total energy} {Oxs_CGEvolve::Delta E} {Oxs\
_CGEvolve::Bracket count} {Oxs_CGEvolve::Line min count} {Oxs_CGEvolve::Conjugate cycle count}\
 {Oxs_CGEvolve::Cycle count} {Oxs_CGEvolve::Cycle sub count} {Oxs_CGEvolve::Energy calc count}\
 Oxs_UniaxialAnisotropy::Energy Oxs_UniformExchange::Energy {Oxs_UniformExchange::Max Spin Ang\
} {Oxs_UniformExchange::Stage Max Spin Ang} {Oxs_UniformExchange::Run Max Spin Ang} Oxs_Demag:\
:Energy    Oxs_MinDriver::Iteration {Oxs_MinDriver::Stage iteration} Oxs_MinDriver::Stage Oxs_\
MinDriver::mx    Oxs_MinDriver::my    Oxs_MinDriver::mz
# Units:              A/m                           J                          J              \
         {}                            {}                               {}                    \
             {}                            {}                            {}                   \
               J                             J                             deg                \
                     deg                                       deg                          J \
                      {}                            {}                           {}           \
   {}                    {}                    {}
             0.00097778028256529097          3.2257415663518404e-16      0                    \
       353                            326                              7                      \
           340                            333                           680                   \
             5.4172367330709765e-17          1.780007679106069e-16           11.011344278380\
658                    90.000000000000014                       90.000000000000014            \
  9.0401021393867362e-17    670                          670                        0          \
           0.40912126717720015  4.6139202285652408e-17 -1.9801501518039851e-16
# Table End



-:---F1  sp3-vortex-seed0000.odt    All L1     (Fundamental)-----------------------------------
File mode specification error: (error "Buffer format not recognized")
```

REPEAT STEPS 1, 2, 3

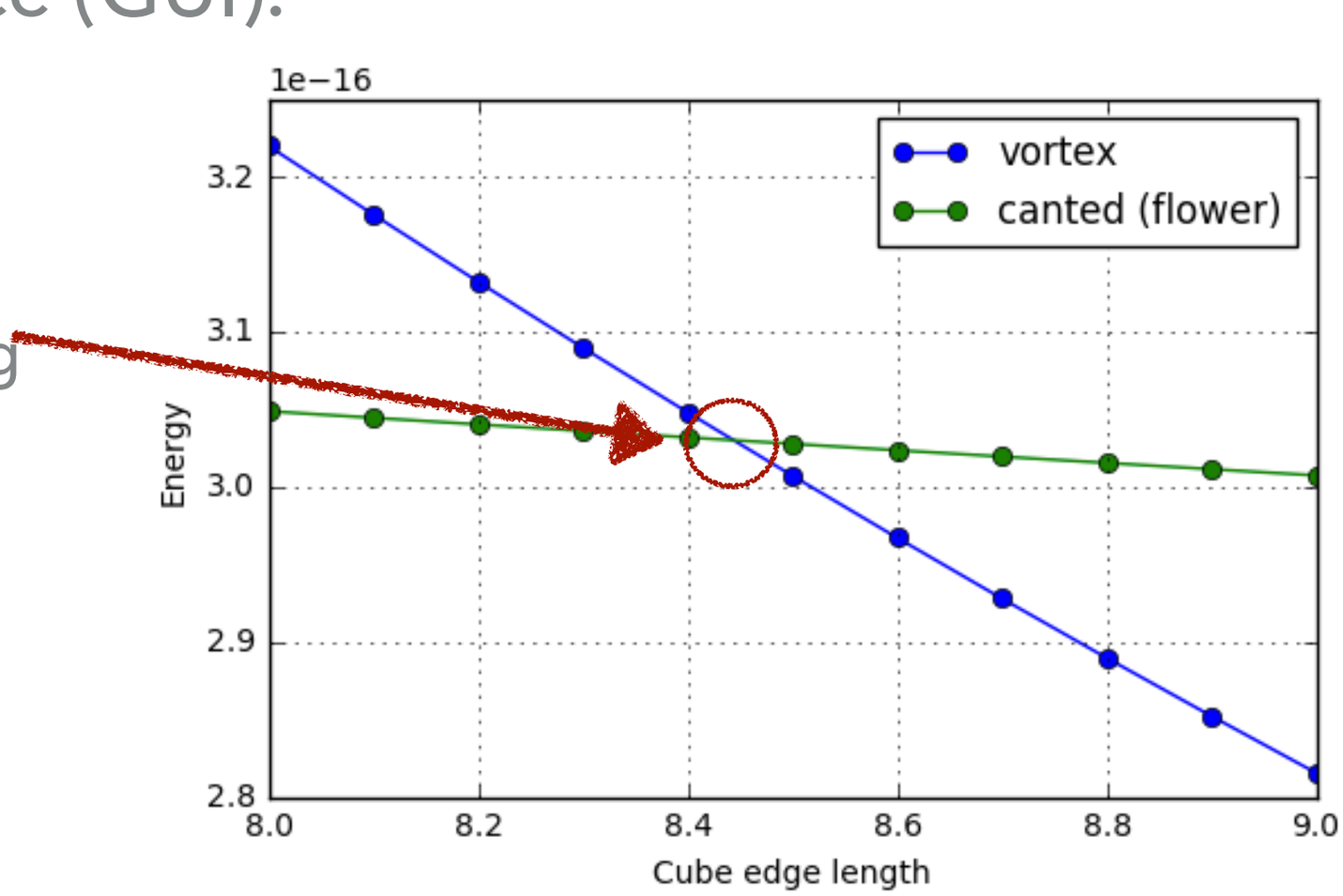
L	flower	vortex
8.0	?	3.23×10^{-16}
8.1	?	?
8.2	?	?
8.3	?	?
8.4	?	?
8.5	?	?
8.6	?	?
8.7	?	?
8.8	?	?
8.9	?	?
9.0	?	?



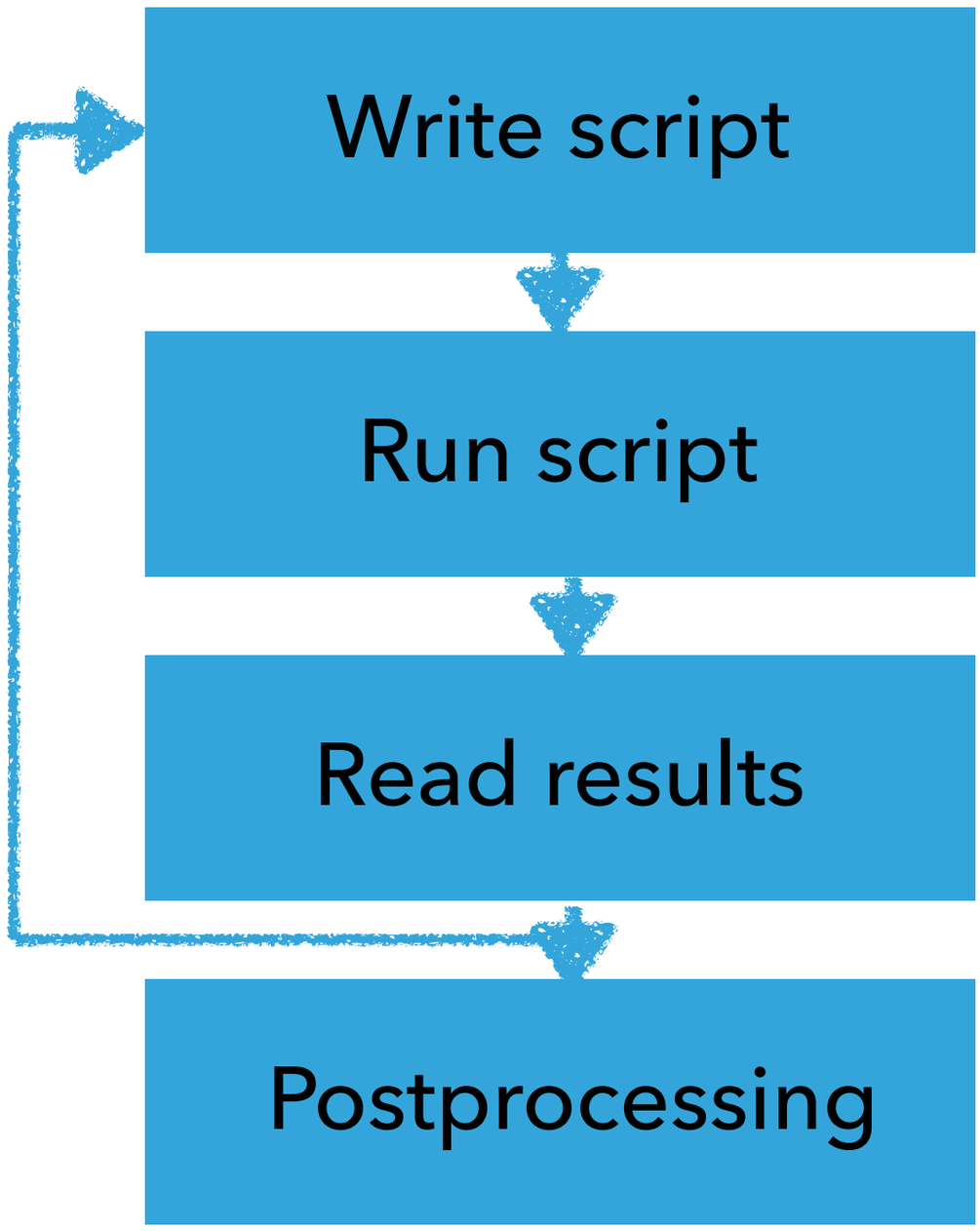
STEP 4: POSTPROCESSING

- Finally, we plot the data and find the crossing.
- We use separate plotting scripts or graphical user interface (GUI).

Find crossing (by eye)



WORKFLOW



EFFECTIVE?

- Time consuming
- Many manual steps (or coding of text-processing scripts)
- Difficult to re-execute automatically
- Difficult to reproduce

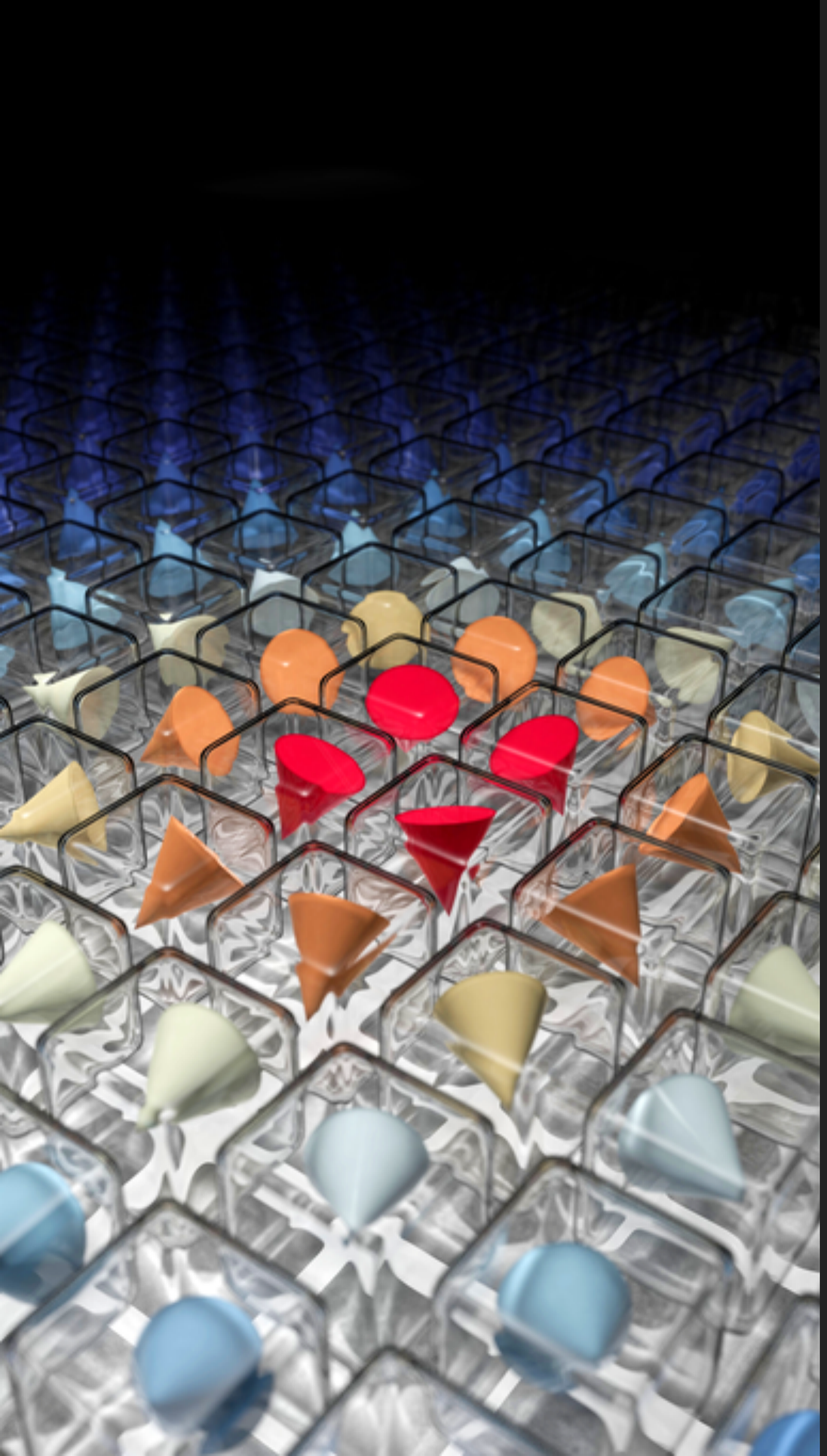
OBJECTIVE: DRIVE THE WHOLE STUDY FROM JUPYTER NOTEBOOK

1. Provide Python interface
2. Integrate with Jupyter Notebook

Why Python?

- Popular language with engineers and scientists [1]
- Many libraries that can be exploited

[1] H. Fangohr. A Comparison of C, Matlab and Python as Teaching Languages in Engineering, *Lecture Notes on Computational Science* **3039**, 1210-1217 (2004).



PART4

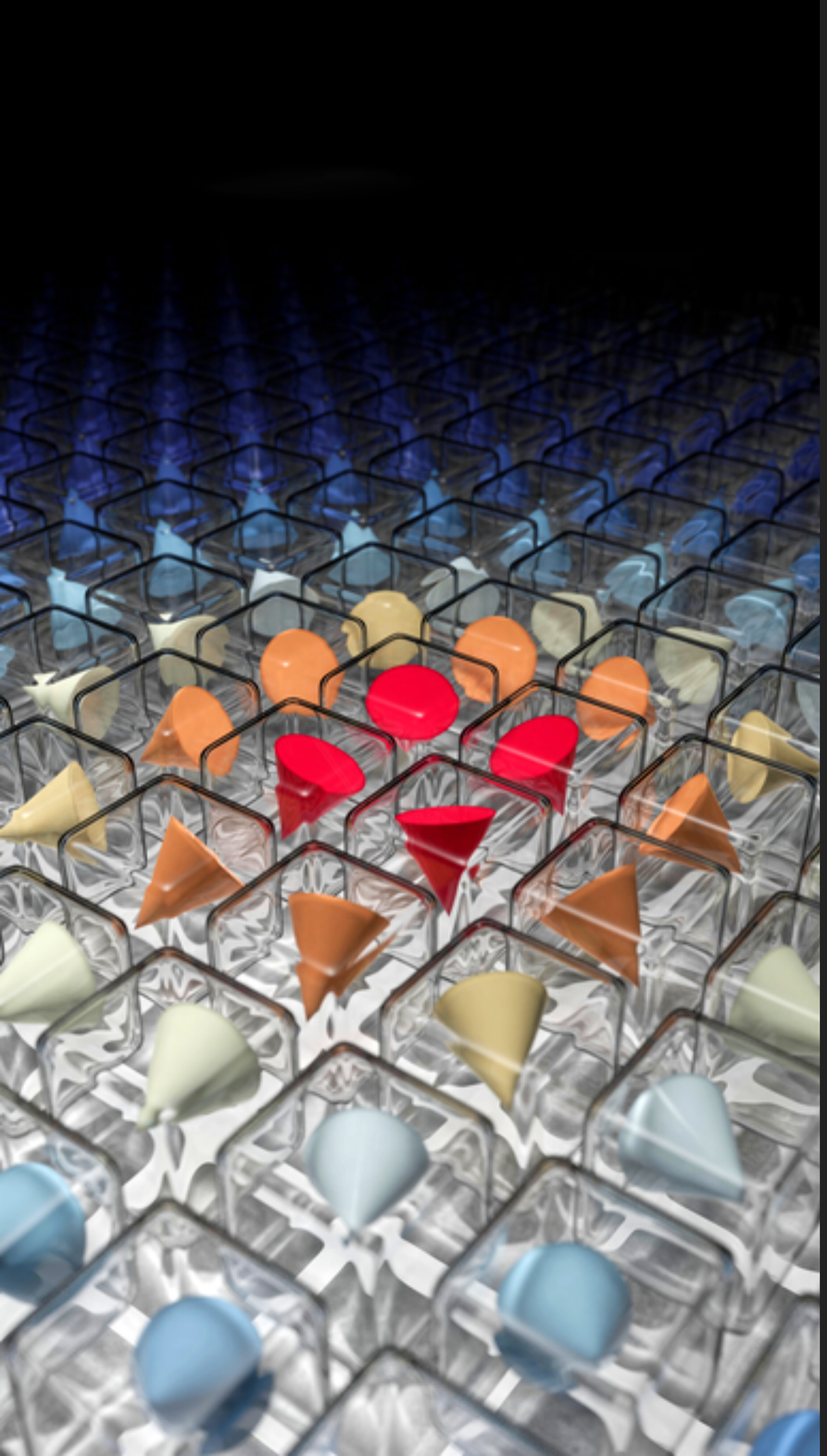
PYTHON INTERFACE

COMMUNICATION WITH OOMMF

- Options include
  - linking to C++ code
  - or communicating with OOMMF via configuration files and data file (most robust solution)

REPRESENTATION OF SIMULATION IN PYTHON

- Micromagnetic model (problem description)
  - Describes the physics, without knowing how to solve it
    - Contains abstract fields
- Micromagnetic calculator (can compute micromagnetic problem)
  - Inherits from micromagnetic model
  - Needs spatial discretisation



PART 5

JUPYTER OOMMF DEMO

JOOMMF INSTALLATION

- Use conda (`conda-forge`)
  - `conda install –channel conda-forge joommf`
  - Provides OOMMF and all dependencies
- Supported on all major operating systems: Windows, MacOS, Linux
- Or install from source or PyPI. In that case we need OOMMF
  - Either installed natively (OOMMF + path),
  - or install Docker (and JOOMMF runs OOMMF in Docker)

ALTERNATIVELY . . . JOOMMF IN THE CLOUD

- JupyterHub instance, behaves like Binder
  - https://tryjoommf.soton.ac.uk
- MyBinder for demo:

For upcoming reproduce-along-session in 10 minutes, open browser at
https://tinyurl.com/yapnotyb

https://mybinder.org/v2/gh/fangohr/talk-pyconde-2018/master

DEMO 1: FIRST STEPS

first-steps.ipynb | standard_problem4.ipynb | standard_problem3.ipynb

Code

Simple micromagnetic problem

```
[ ]: %matplotlib inline
import oommfc as oommf_calc
import discretisedfield as df

system = oommf_calc.System(name="name")
```

We define the mesh by providing two points between which the domain spans as well as the size of the discretisation cell.

```
[ ]: L = 100e-9     # m ; edge length of simulation cube
d = 10e-9      # m ; edge length of discretisation cell size
mesh = oommf_calc.Mesh(p1=(0, 0, 0), p2=(L, L, L), cell=(d, d, d))
```

```
[ ]:
```

Our Hamiltonian contains only exchange, demagnetisation, and Zeeman energy terms.

```
[ ]: H = (8e6, 0, 0)    # apply field in x-direction
system.hamiltonian = oommf_calc.Exchange(A=1e-12) + oommf_calc.Demag() + oommf_calc.Zeeman(H=H)
```

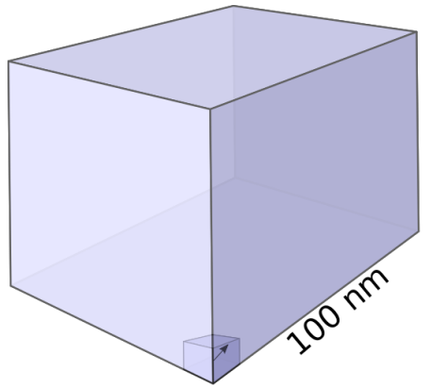
```
[ ]:
```

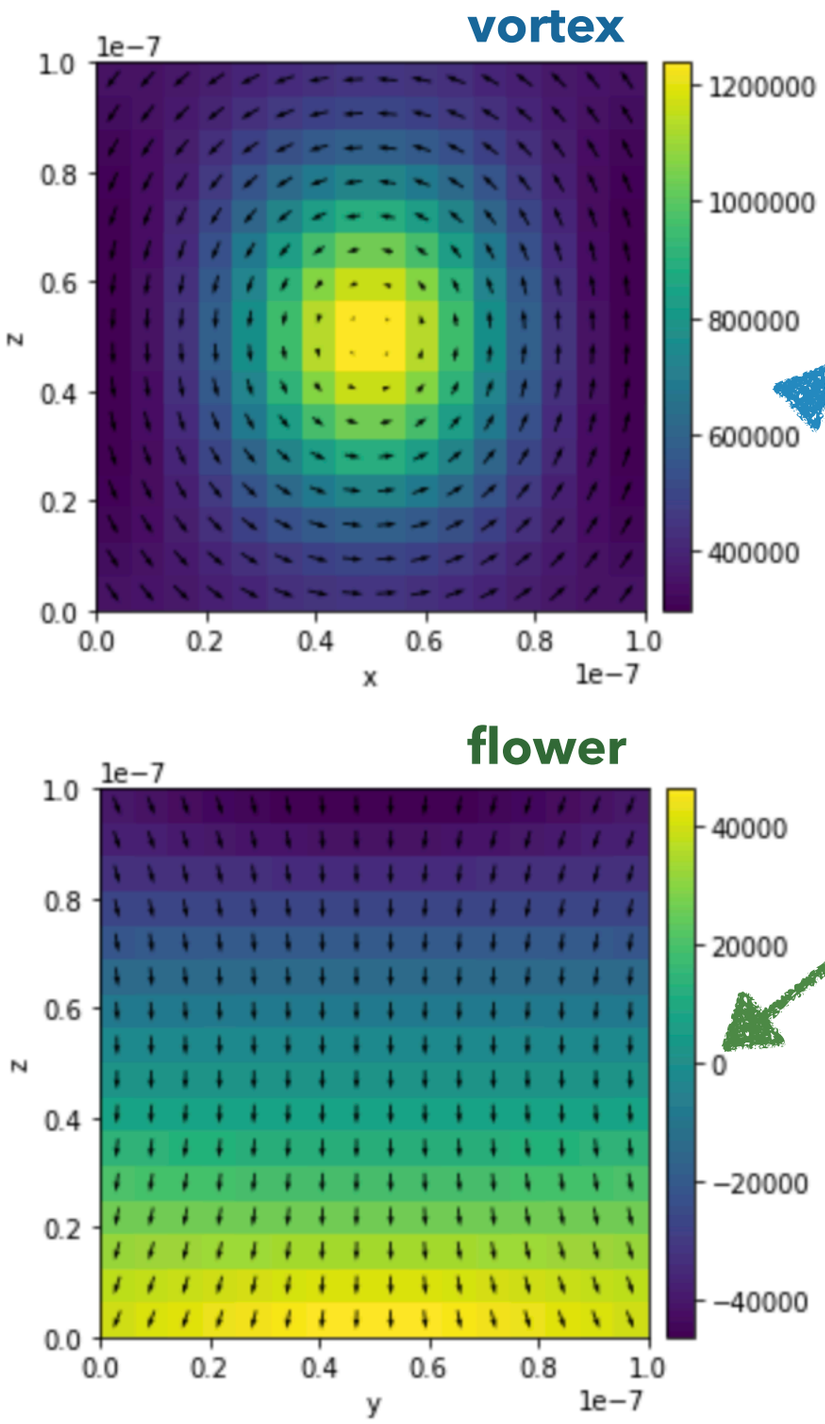
Dynamics of the system is governed by the LLG equation containing precession and damping terms.

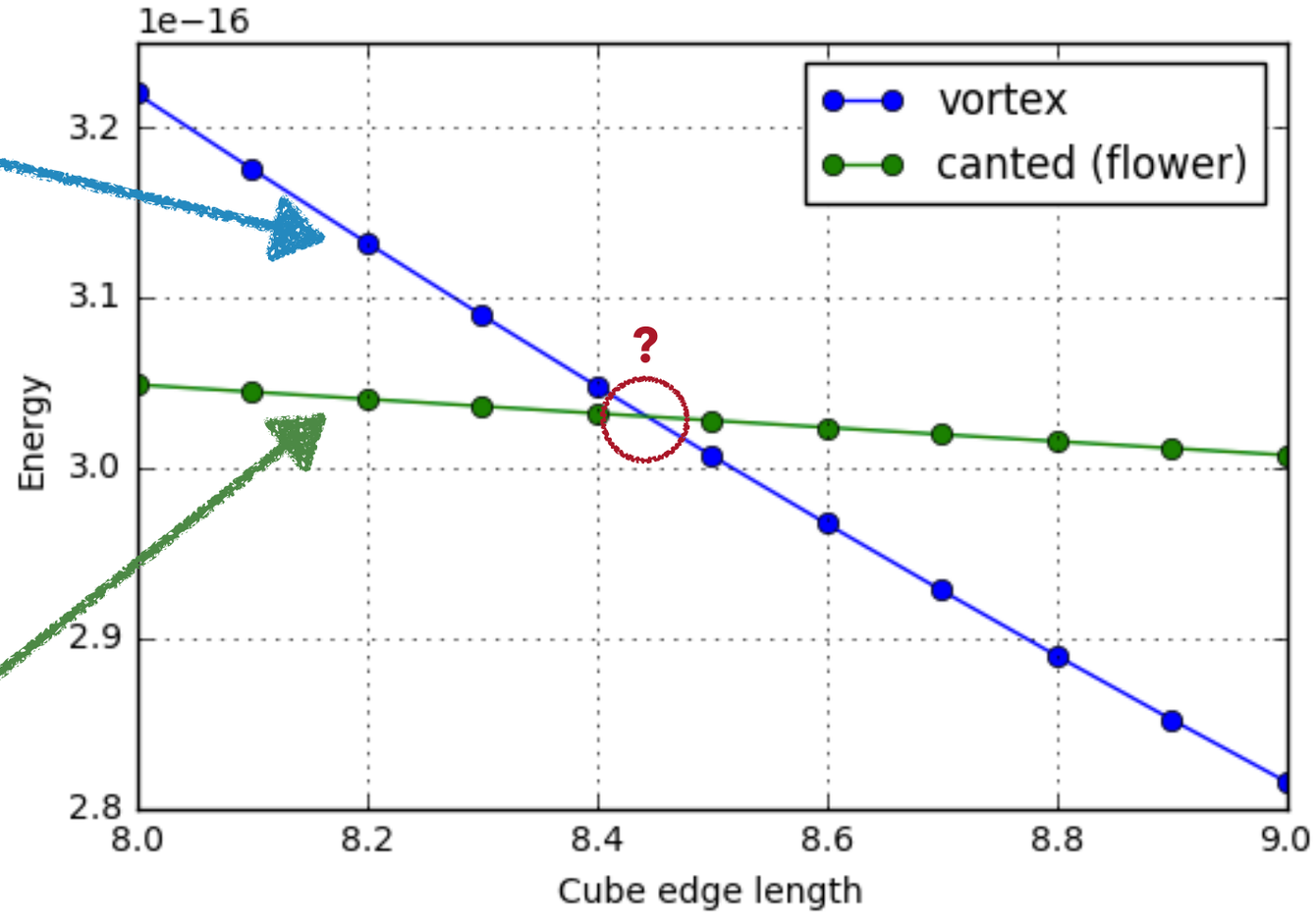
```
[ ]: system.dynamics = oommf_calc.Precession(gamma=oommf_calc.gamma0) + oommf_calc.Damping(alpha=0.2)
```

```
[ ]:
```

DEMO 2: STANDARD PROBLEM 3







DEMO 3: TIME-DEPENDENT PROBLEM

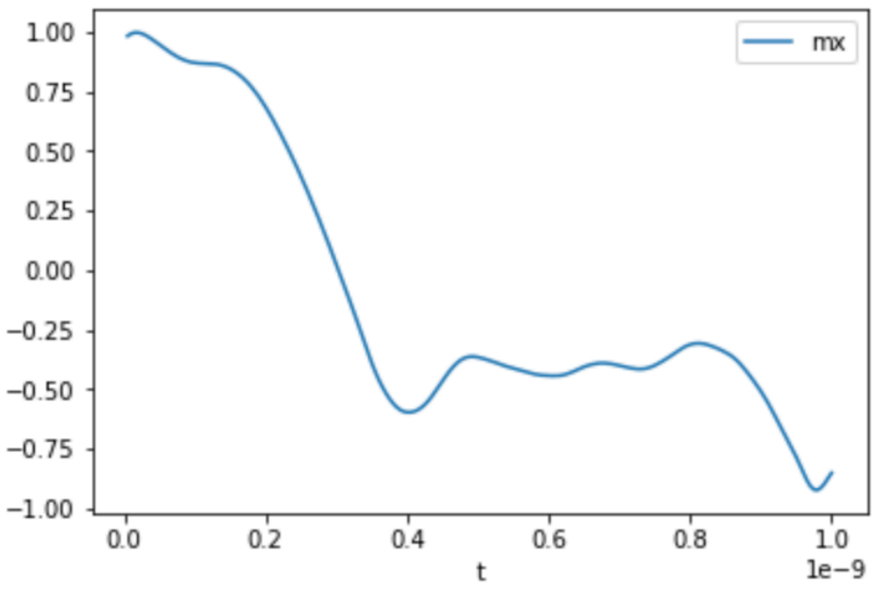
[31]:
```
system.dt.tail()
```

[31]:

	E	Ecount	max_dm/dt	dE/dt	deltaE	E_Exchange	max_spin_angle	stage_max_spin_angle	run_max_spin_angle
195	2.112534e-18	3748.0	5298.625106	-9.295965e-09	-1.266227e-20	1.742733e-18	53.666961	53.666961	53.666961
196	2.067159e-18	3767.0	5035.192967	-8.815293e-09	-1.204369e-20	1.874195e-18	51.404011	53.818936	53.818936
197	2.024655e-18	3786.0	6066.902182	-8.176785e-09	-1.117432e-20	2.012887e-18	44.249806	51.404011	53.818936
198	1.985246e-18	3805.0	6124.734285	-7.629952e-09	-1.034672e-20	2.146091e-18	48.942501	48.942501	53.818936
199	1.947593e-18	3824.0	6004.232063	-7.542039e-09	-1.007869e-20	2.267998e-18	51.818155	52.056473	53.818936

Finally, we want to plot the average magnetisation configuration `my` as a function of time `t`:

[39]:
```
myplot = system.dt.plot("t", "mx")
```



ALL OPEN SOURCE

https://github.com/joommf/joommf

joommf

Marijan Beg[1,2], Ryan A. Pepper[2], Thomas Kluyver[1], and Hans Fangohr[1,2]

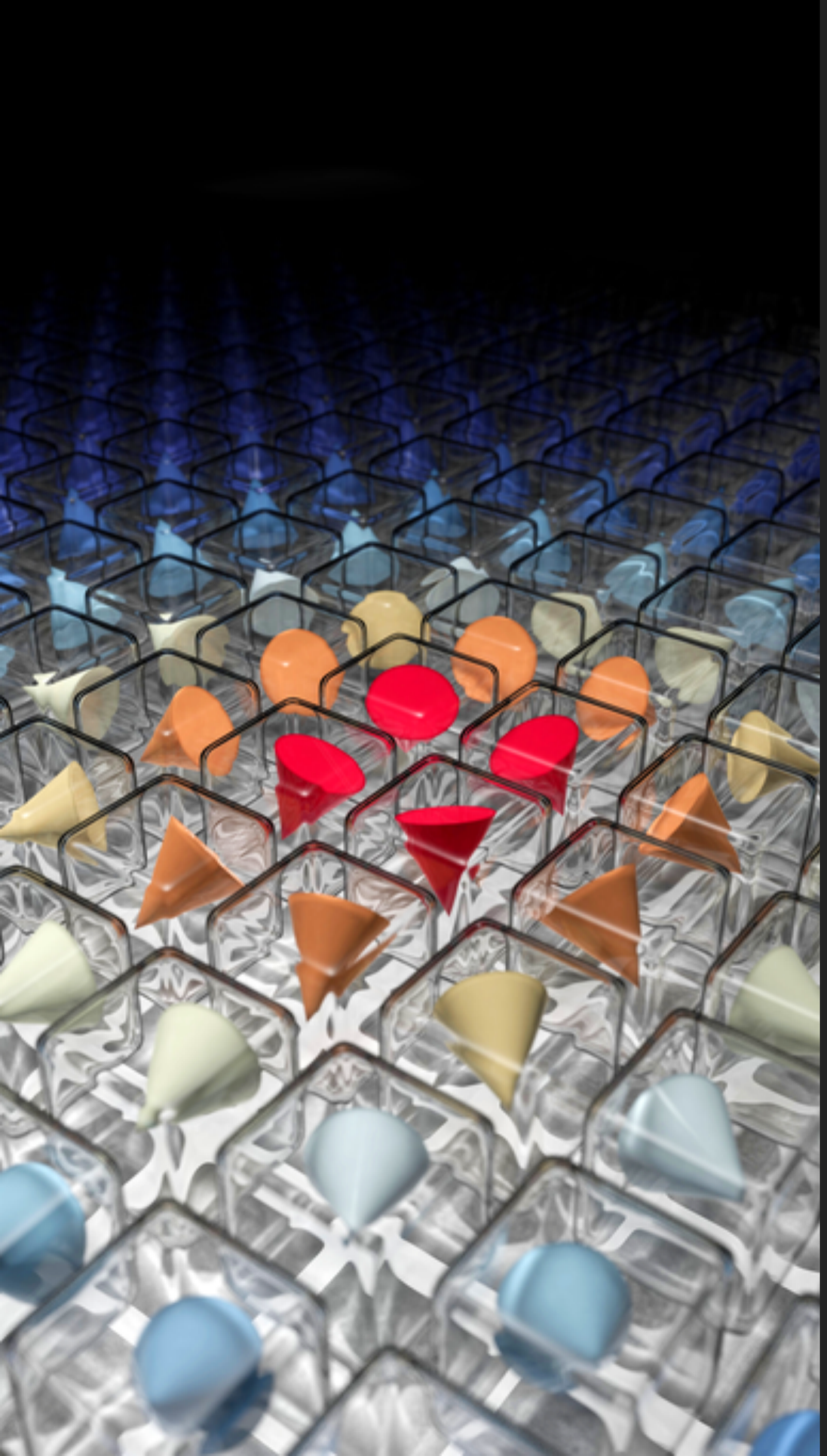
[1] *European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, Germany*
[2] *Faculty of Engineering and the Environment, University of Southampton, Southampton SO17 1BJ, United Kingdom*

Description	Badge
Latest release	pypi package 0.8.4
	anaconda cloud 0.8.4
Build	build passing
	build passing
Coverage	codecov 100%
Documentation	docs unknown
Binder	launch binder
Dependecies	requirements up-to-date
License	license BSD 3-Clause

PACKAGES

Dependencies overview

	joommfutil	discretisedfield	oommfodt	micromagneticmodel	oommfc
Latest release	pypi package 0.8.3	pypi package 0.8.3	pypi package 0.8.2	pypi package 0.8.4	pypi package 0.8.8
	anaconda cloud 0.8.3	anaconda cloud 0.8.3	anaconda cloud 0.8.2	anaconda cloud 0.8.2	anaconda cloud 0.8.8
Build	build passing	build passing	build passing	build passing	build passing
	build passing	build passing	build passing	build passing	build passing
Coverage	codecov 98%	codecov 95%	codecov 94%	codecov 100%	codecov 92%
Documentation	docs passing	docs passing	docs passing	docs passing	docs passing
Binder	launch binder	launch binder	launch binder	launch binder	launch binder
Dependecies	requirements up-to-date	requirements up-to-date	requirements up-to-date	requirements up-to-date	requirements up-to-date
License	license BSD 3-Clause	license BSD 3-Clause	license BSD 3-Clause	license BSD 3-Clause	license BSD 3-Clause



PART 6

DISCUSSION AND SUMMARY

WHAT HAVE WE GOT?

- Ability to drive micromagnetic simulations (using OOMMF) from Python
- Integration with Jupyter Notebook
  - Rich media representation of equations, meshes, fields
  - Widgets to explore data sets interactively in notebook
- Framework to include more micromagnetic computational solvers (for example mumax3, micromagnum, fidimag)

WHAT ELSE DO WE GET?

- Scriptability of computational studies within Python
- Use of the Python ecosystem for computational and data science, including numpy, scipy, pandas, ...
- Easier reproducibility: Notebook contains complete simulation study
- Sharing of interactive documents through MyBinder
- Remote execution of simulation (JupyterHub)

WHAT ELSE DO WE GET? (2)

- Documentation and Tutorials in Notebooks
  - Sphinx docs created from this
  - Users can download notebooks and execute/modify
- Users can execute/modify notebooks on the web (MyBinder)
- Can use continuous integration on documentation notebooks
  - py.test and nbval (https://github.com/computationalmodelling/nbval)

```
937 $ make test-ipynb
938 mkdir -p test-reports/junit
939 cd doc/ipynb && py.test . -v --nbval --
    junitxml=/home/travis/build/computationa
940 ============================= test
941 platform linux -- Python 3.5.2, pytest-
    test/bin/python
942 cachedir: ../../.cache
943 rootdir: /home/travis/build/computation
944 plugins: cov-2.3.1, nbval-0.3.6
945 collected 65 items
946
947 1d_domain_wall.ipynb::Cell 5 PASSED
948 1d_domain_wall.ipynb::Cell 7 PASSED
949 1d_domain_wall.ipynb::Cell 9 PASSED
950 1d_domain_wall.ipynb::Cell 11 PASSED
951 1d_domain_wall.ipynb::Cell 13 PASSED
952 1d_domain_wall.ipynb::Cell 15 PASSED
953 current-driven-domain-wall.ipynb::Cell
954 current-driven-domain-wall.ipynb::Cell
955 current-driven-domain-wall.ipynb::Cell
956 current-driven-domain-wall.ipynb::Cell
957 current-driven-domain-wall.ipynb::Cell
958 current-driven-domain-wall.ipynb::Cell
959 current-driven-domain-wall.ipynb::Cell
960 current-driven-domain-wall.ipynb::Cell
961 current-driven-domain-wall.ipynb::Cell
962 current-driven-domain-wall.ipynb::Cell
963 current-driven-domain-wall.ipynb::Cell
964 isolated_skyrmion.ipynb::Cell 5 PASSED
```

COMPUTER SCIENCE PERSPECTIVE ON THE USER INTERFACE

- Python libraries created are a Domain Specific Language (DSL) for micromagnetic science
- This DSL is embedded in general purpose programming language (Python)
  - More powerful than (i) hard coded parameters, or (ii) config files
  - But also high complexity: users can combine library functions in all possible ways
- Publication: M. Beg, R. A. Pepper, and H. Fangohr. User interfaces for computational science: A domain specific language for OOMMF embedded in Python. *AIP Advances* **7**, 56025 (2017). https://doi.org/10.1063/1.4977225

USER FEEDBACK FROM SHORT WORKSHOPS

- Scientists *without programming experience* struggle with Python in Notebook setup: many new concepts at the same time
- Generally well received
- JOOMMF in the cloud (JupyterHub, MyBinder) very effective for workshop delivery


ACKNOWLEDGEMENTS

- Financial support:
  - OpenDreamKit Horizon 2020, European Research Infrastructures project (#676541), http://opendreamkit.org,
  - EPSRC's Skyrmion Programme Grant (EP/N032128/1),
  - EPSRC's Centre for Doctoral Training in Next Generation Computational Modelling, http://ngcm.soton.ac.uk (#EP/L015382/1), and
  - The Gordon and Betty Moore Foundation through Grant GBMF #4856, by the Alfred P. Sloan Foundation and by the Helmsley Trust.

SUMMARY

- Jupyter-OOMMF: joommf.github.io
- Contributors: Marijan Beg, Sergii Mamedov, Ryan A. Pepper, David Cortés-Ortuño, Thomas Kluyver, Hans Fangohr
- Publication: M. Beg, R. A. Pepper, and H. Fangohr. User interfaces for computational science: A domain specific language for OOMMF embedded in Python. *AIP Advances* **7**, 56025 (2017). https://doi.org/10.1063/1.4977225
- This talk, and link to MyBinder execution of notebooks https://github.com/fangohr/talk-pyconde-2018
- Open positions and contact: hans.fangohr@xfel.eu, @ProfCompMod, fangohr.github.io