

Jupyter-OOMMF: VRE for Micromagnetic Simulations

Marijan Beg,^a Ryan A. Pepper,^a Leoni Breth,^a Vidar T Fauske,^b Benjamin Ragan-Kelley,^b Hans Fangohr^a
 a) University of Southampton, Southampton, United Kingdom
 b) Simula Research Laboratory, Norway

Introducing Jupyter-OOMMF

- provides a user-friendly interface to the finite difference code **OOMMF** (<http://math.nist.gov/oommf/>)
- enables the use of **Jupyter notebooks** (jupyter.org), which can be run in any web browser (illustrated in Fig. 1)
- commands allow to run a full simulation workflow within a single notebook instead of using different tools in each step of the workflow (see Fig. 2)
- interaction of the frontend layer written in **Python 3** with OOMMF through `.mif`-configuration files¹
- extendable to other code packages in the future
- freely available on github (<https://github.com/joommf>)
- website with documentation: joommf.github.io



Fig. 1: JOOMMF = Jupyter + OOMMF

Standard problem: find critical size of magnetic cube [2]

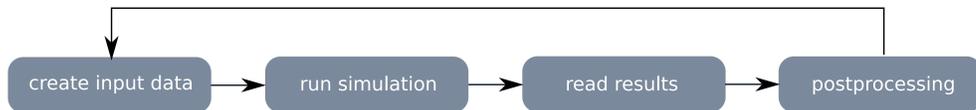


Fig. 2: Simulation workflow

function, that returns the relaxed system object, argument L is the cube length in units of lex

the "system" object contains a set of properties, such as "hamiltonian" and "m"

here, the OOMMF configuration file (*.mif) is created for the energy minimization

energy minimization is executed starting from the vortex state

energy minimization is executed starting from the flower state

Energy crossing

Now, we can plot the energies of both vortex and flower states as a function of cube edge length. This will give us an idea where the state transition occurs.

for-loop to step through the values for the cube length L and compute the corresponding energies

Python's "scipy" module provides a wide range of useful algorithms for further data evaluation

[1] Beg, M., Pepper R., Fangohr H., AIP Advances 7, 056025 (2017)
 [2] mumag Standard Problem 3: www.ctcms.nist.gov/~rdm/mumag.org.html