

WARNING: This is a presentation PDF and does not have animations. For a MacOS Keynote presentation, ask Marijan.

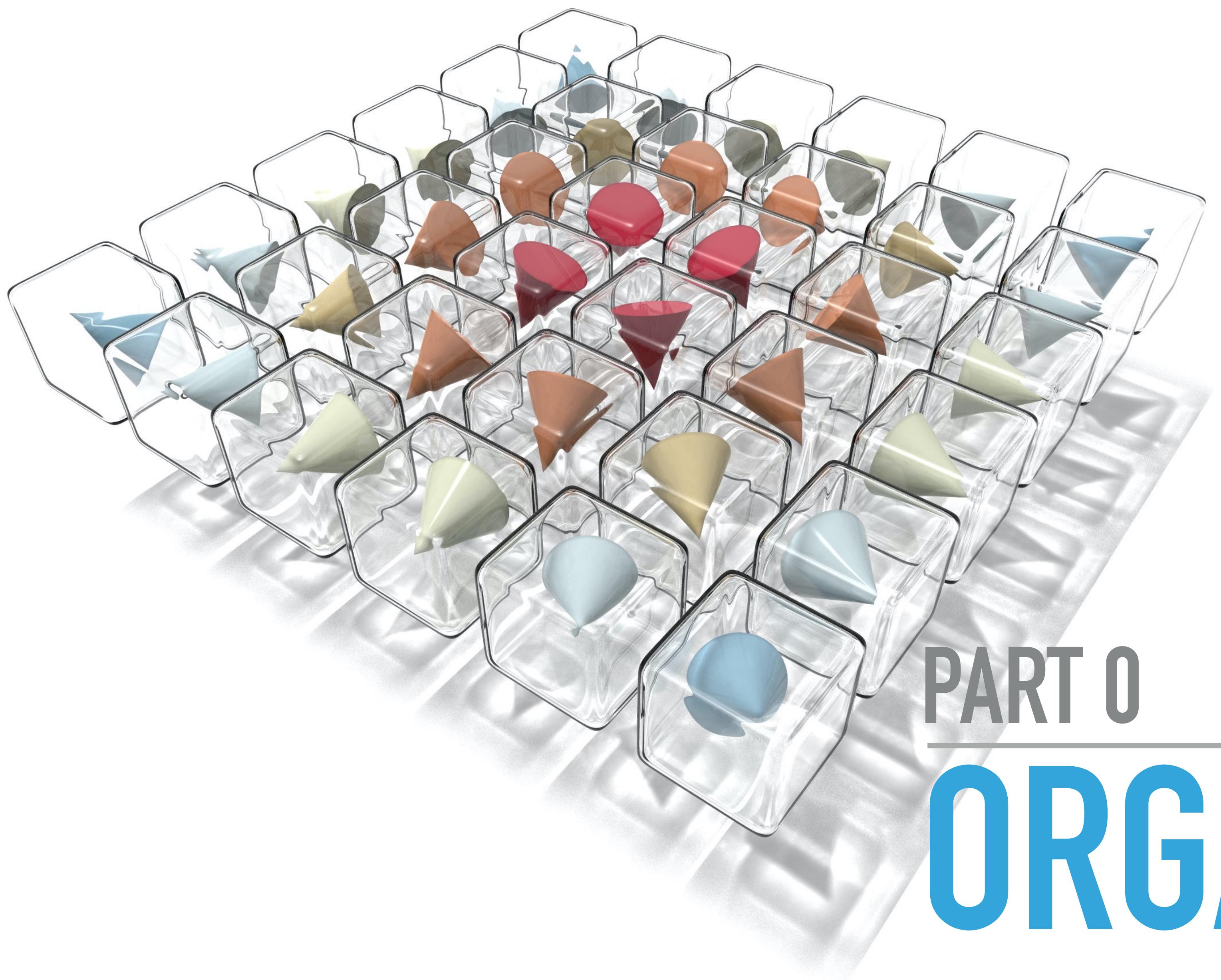


Marijan Beg¹, Ryan A. Pepper¹, and Hans Fangohr^{1,2}

MICROMAGNETICS WITH UBERMAG

¹University of Southampton, Highfield, SO17 1BJ Southampton, United Kingdom

²European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, Germany



PART 0

ORGANISATION

ACKNOWLEDGEMENTS

- ▶ We acknowledge the **IEEE Magnetics Society** support.



- ▶ We thank **Xin Fan** and **Kirill Belashchenko** for organising the workshop.

PEOPLE

- ▶ Instructor:
 - ▶ **Marijan Beg** (@marijanbeg)
- ▶ Moderators:
 - ▶ **Ryan A. Pepper** (@rpep)
 - ▶ **Hans Fangohr** (@fangohr)

QUESTIONS

- ▶ There are **two ways how** you can ask a question:
 1. **Raise an issue** in the workshop repository. (Please be aware that questions and answers can be seen by everybody.)
 2. During live sessions, please send your questions as a **private Zoom message** to:
 - ▶ **Ryan A. Pepper**
 - ▶ **Hans Fangohr**
- ▶ If you are on **Twitch**, send your questions there and we are going to find it.
- ▶ Due to the limited time and a large number of participants, **we are not going to be able to answer all questions.**
 - ▶ If you do not get an answer during the live session, please raise an issue in the workshop repository.

SESSIONS

- ▶ The workshop is divided into **3 sessions**:
 1. (DONE) Basics of micromagnetics, Jupyter, and Python
 - ▶ **Thursday 18 June 2020, 12:00-13:30 (ET)**
 2. (NEXT) Micromagnetic models and drivers
 - ▶ **Thursday 25 June 2020, 12:00-13:30 (ET)**
 3. Data analysis and visualisation
 - ▶ **Thursday 02 June 2020, 12:00-13:30 (ET)**
- ▶ For each session we only imply what the main focus is going to be.
 - ▶ In each session we are going to be covering all the topics, but at different levels.

AIMS

- ▶ **Survey:** The majority of participants are PhD students and most people never ran a micromagnetic simulation before.
- ▶ Our goal is to try to establish **a good foundation** to make sure that at the end of the workshop you know how to:
 - ▶ **Set up and run** a micromagnetic simulation in ubermag,
 - ▶ **Analyse and visualise** the data,
 - ▶ **Understand the documentation** and learn more advanced topics on your own.

WHAT DO I NEED FOR THE WORKSHOP?

- ▶ The **workshop repository** is going to be updated regularly and it is available at:

<https://github.com/ubermag/workshop>

- ▶ It contains all the slides, tutorials, exercises, extra materials, and up-to-date information about the workshop.
- ▶ Tutorials and exercises can be **run in the cloud** using Binder.
- ▶ You **do not need to install anything** and no files will be created on your machine.
- ▶ Binder can be accessed using **Binder badge** in the workshop repository:



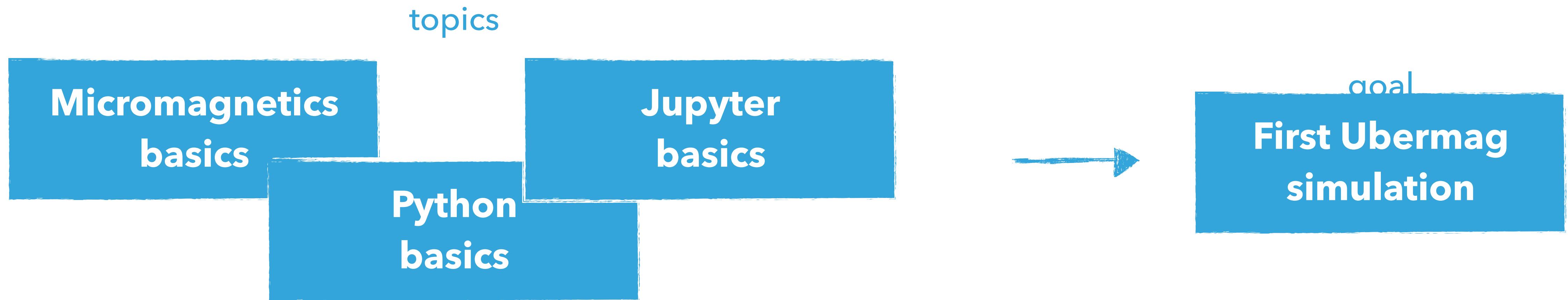
- ▶ Starting Binder can take a few minutes, so **please be patient**.
- ▶ **WARNING:** The most recent update of Safari web browser on MacOS sometimes does not interpret the colours well in 3D interactive plots and **Google Chrome is recommended**.

INSTALLATION (OOMMF VS MUMAX3)

- ▶ Ubermag is a **Python package** and it can be installed in many different ways (pip, miniconda, anaconda-navigator, docker...)
- ▶ We recommend the installation using **miniconda**:
 1. Install Miniconda
 2. conda install --channel conda-forge ubermag
- ▶ For detailed installation instructions, please follow one of our **YouTube videos**.
- ▶ Due to the **limited access to GPU machines**, we are going to be focusing on OOMMF.
 - ▶ By installing ubermag, OOMMF is going to be installed automatically.
- ▶ **Ubermag interface is the same** for both OOMMF and mumax3.
- ▶ mumax3c is currently available from the development branch, and we expect to demonstrate **mumax3c in the last session**. The current situation makes it a bit difficult to make a release.

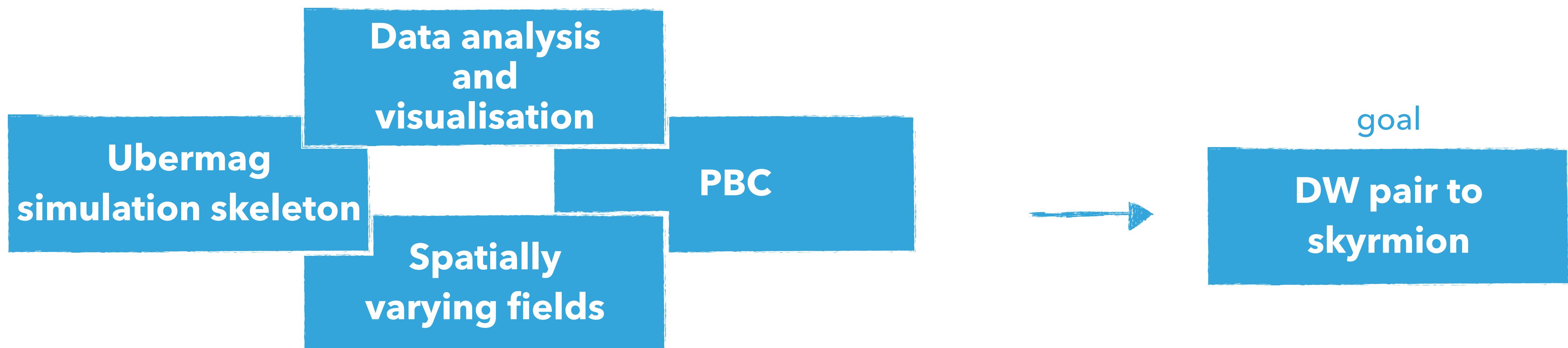
THE PLAN FOR SESSION 1

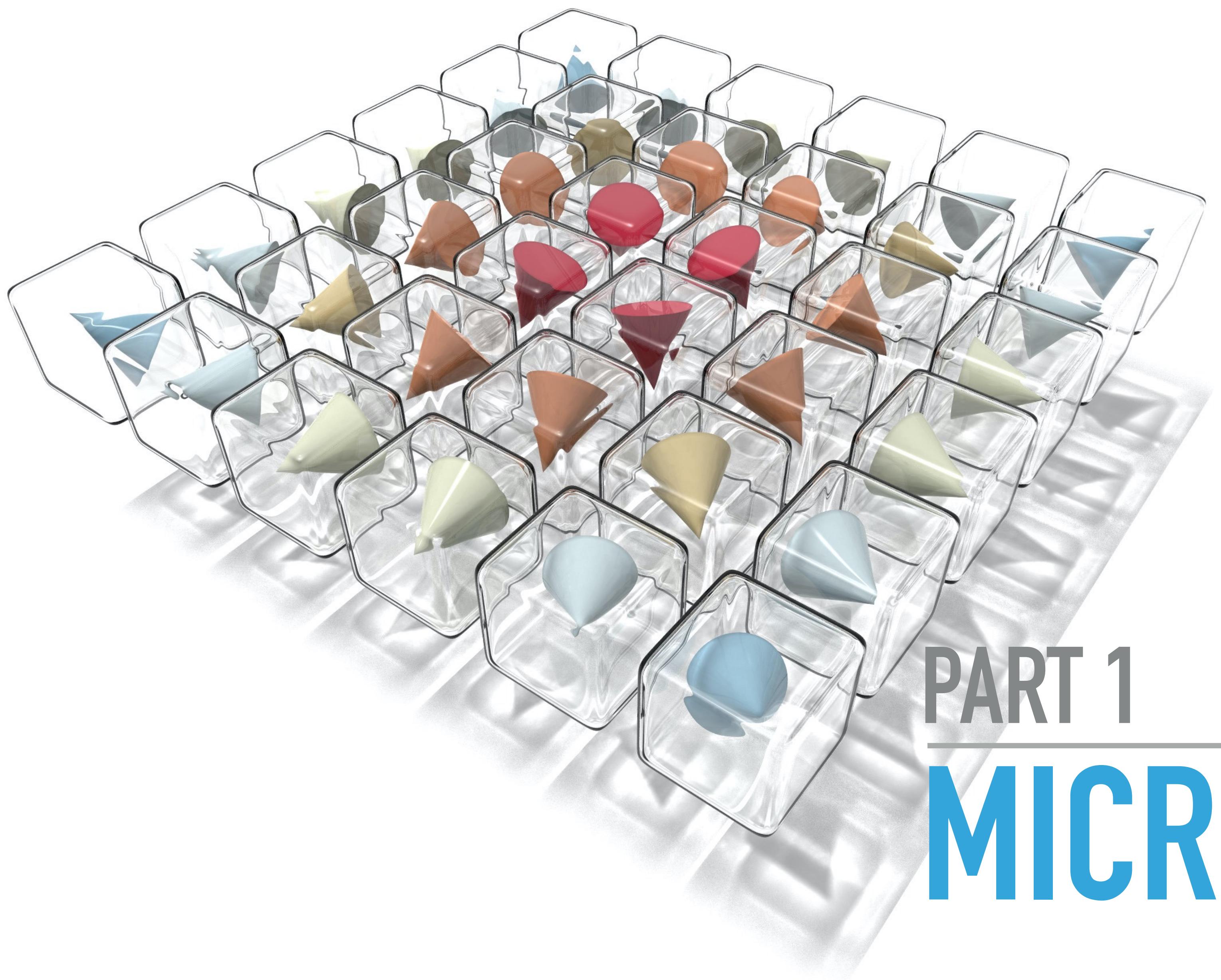
- ▶ The aim of the workshop is to gain some **basic understanding of micromagnetics** and to be **able to run useful simulations** in the end.
- ▶ Our assumption is **no previous experience** in micromagnetics, Python, or Jupyter.
- ▶ In Session 1, we are going to **cover only the very basics**, so that everybody should be able to follow sessions 2 and 3 (the most exciting things happen later).



THE PLAN FOR SESSION 2

- ▶ In Session 2, we build on top of the basics we introduced in session 1.
- ▶ We are going to have a look at spatially varying fields, parameters, energy and dynamics equations, as well as periodic boundary conditions.
- ▶ In addition, we are going to start looking at some data analysis and visualisation methods.





PART 1

MICROMAGNETICS

MICROMAGNETICS

“... is a field of physics dealing with the prediction of magnetic behaviours at sub-micrometer length scales.”

Source: Wikipedia

MAGNETISATION FIELD

- ▶ In continuum approximation, magnetisation is considered to be a **continuous vector field**.
- ▶ Magnetisation $\mathbf{M}(\mathbf{r}, t)$ is a function of both space \mathbf{r} and time t .

$$\mathbf{M} = \mathbf{M}(\mathbf{r}, t)$$

MICROMAGNETIC ASSUMPTIONS

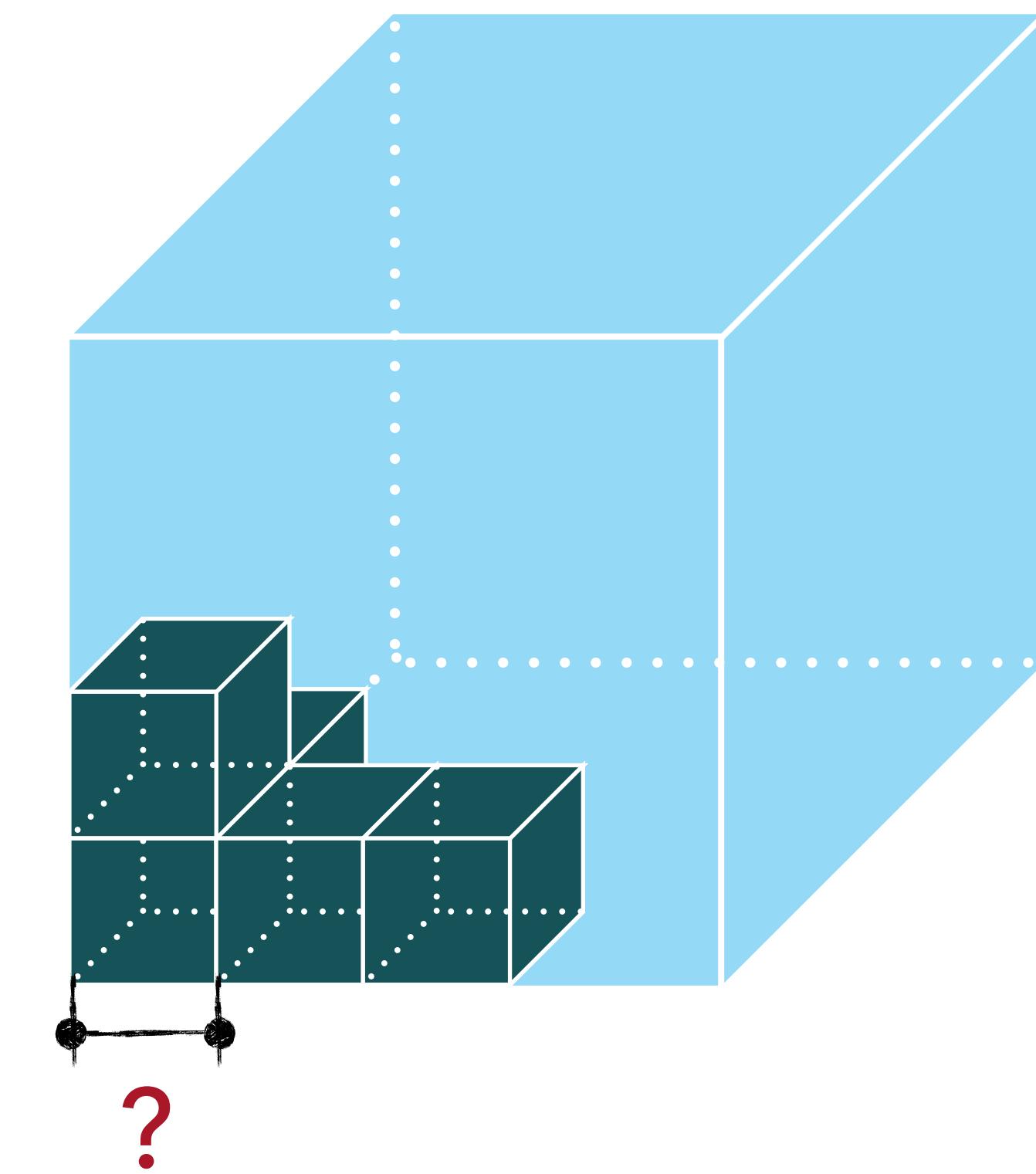
1. Magnetisation field $\mathbf{M}(\mathbf{r}, t)$ is **differentiable** (continuous and slowly changing) with respect to both space \mathbf{r} and time t .
2. The magnetisation field **norm is constant** (time invariant).
 - ▶ Constant norm $|\mathbf{M}|$ is represented by **saturation magnetisation** M_s .
$$M_s = |\mathbf{M}| = \text{const.}$$
 - ▶ Very often, magnetisation is represented by a normalised magnetisation field $\mathbf{m}(\mathbf{r}, t)$.

$$\mathbf{m}(\mathbf{r}, t) = \frac{\mathbf{M}(\mathbf{r}, t)}{M_s}$$

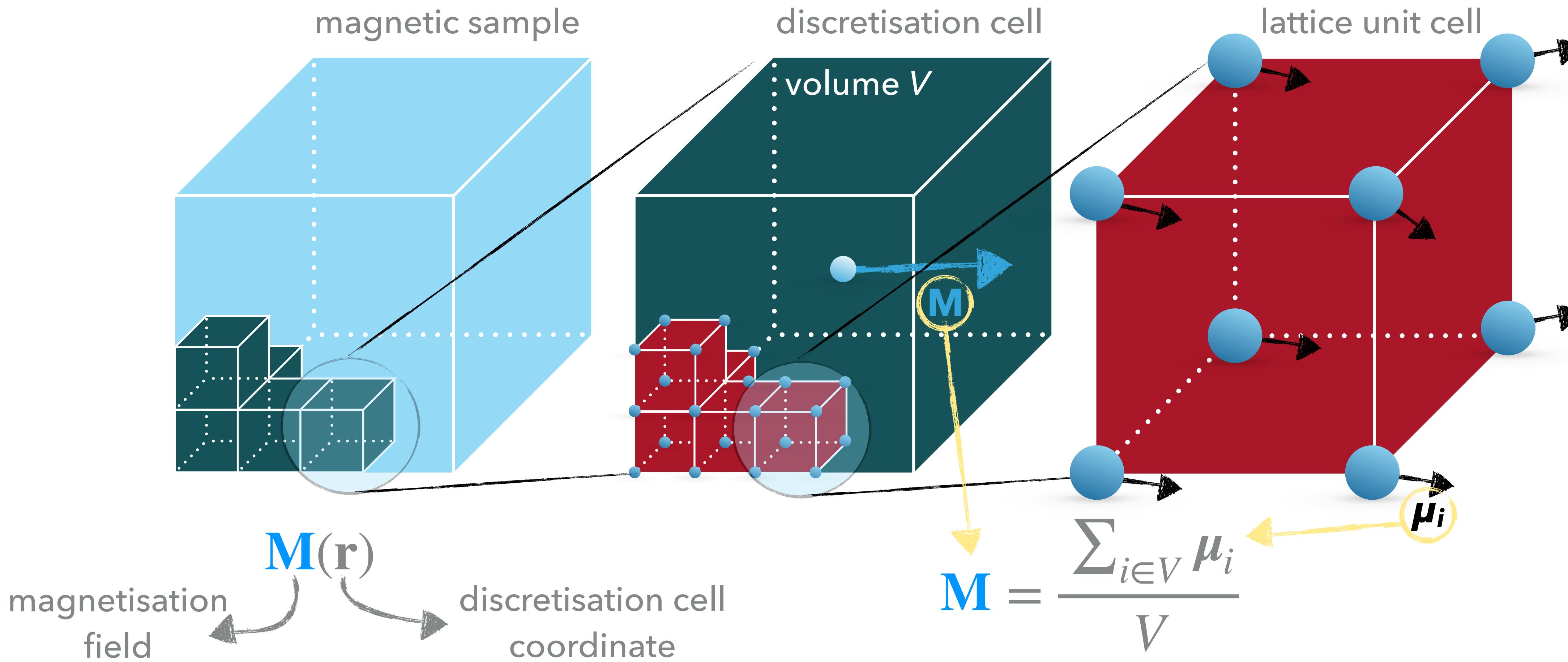
$$|\mathbf{m}| = m_x^2 + m_y^2 + m_z^2 = 1$$

DISCRETISATION

- ▶ In order to solve the magnetisation field numerically, we have to divide it into smaller “chunks”.
- ▶ There are two main ways how we can discretise the field:
 - ▶ **finite-differences**
 - ▶ finite-elements
- ▶ The discretisation must be:
 - ▶ **large enough** to ignore the crystal structure of the material (the continuum approximation).
 - ▶ **small enough** to spatially resolve different magnetisation configurations (like domain walls, vortices, or skyrmions).



FINITE-DIFFERENCE DISCRETISATION



DISCRETISED MAGNETISATION FIELD

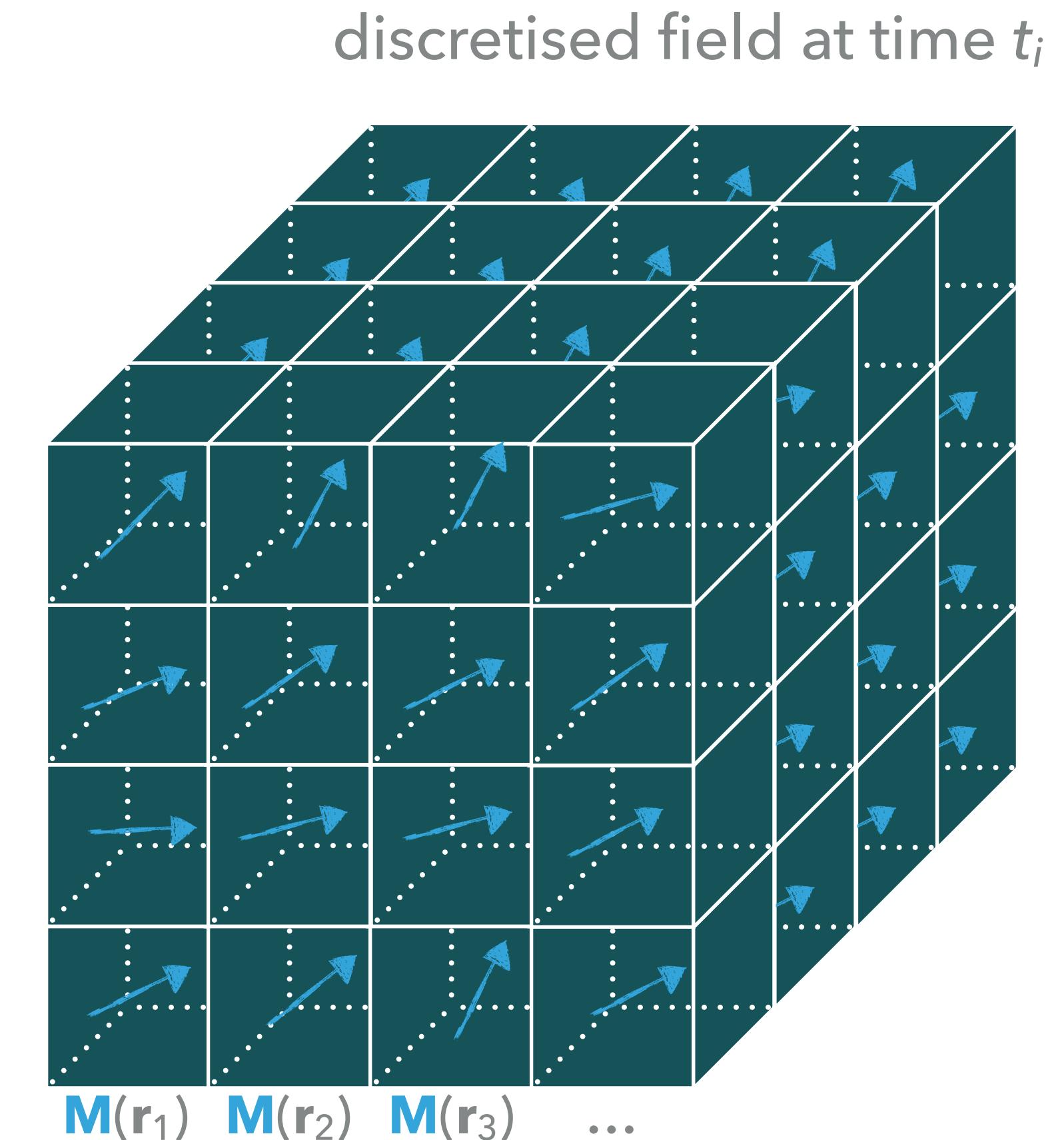
- In continuum approximation, magnetisation is considered to be a **continuous vector field**.

- Magnetisation $\mathbf{M}(\mathbf{r}, t)$ is a function of both space \mathbf{r} and time t .

$$\mathbf{M} = \mathbf{M}(\mathbf{r}, t)$$

- In finite-differences, magnetisation field \mathbf{M} is discretised (at every time step) so that a **single vector is assigned to each discretisation cell**.

$$\mathbf{M}(\mathbf{r}_i, t_i) = (M_x, M_y, M_z)$$



ENERGY EQUATION (HAMILTONIAN)

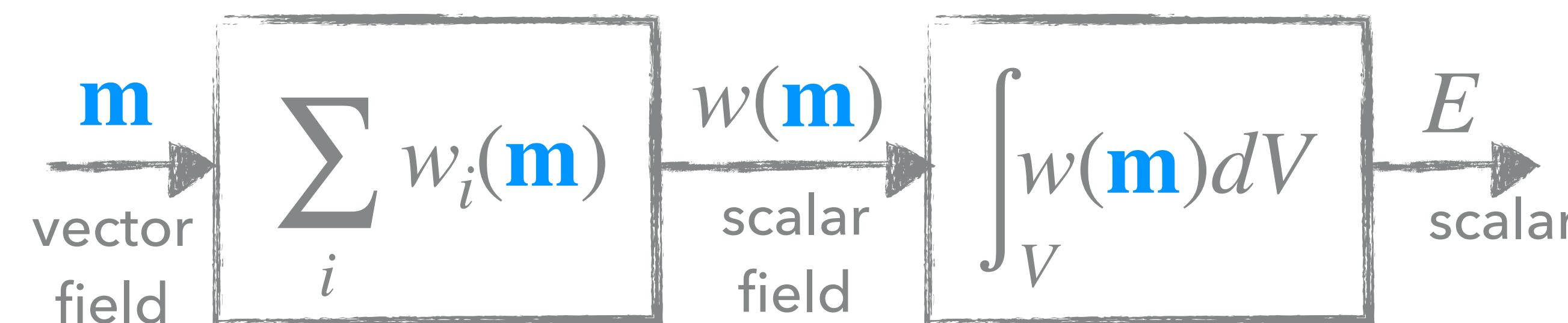
- ... is a mapping of magnetisation field $\mathbf{m}=\mathbf{m}(\mathbf{r}, t)$ to energy density (scalar) field.

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

user-defined

- By integrating $w(\mathbf{m})$ over the entire sample volume V , the energy functional is

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

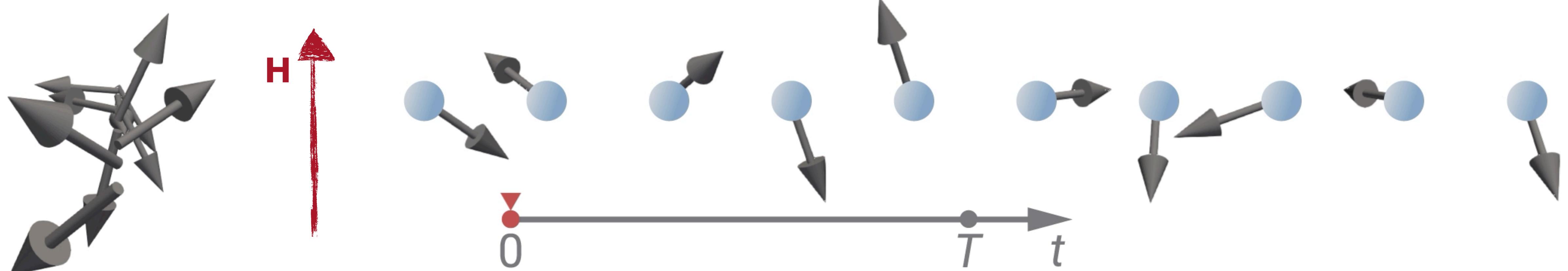
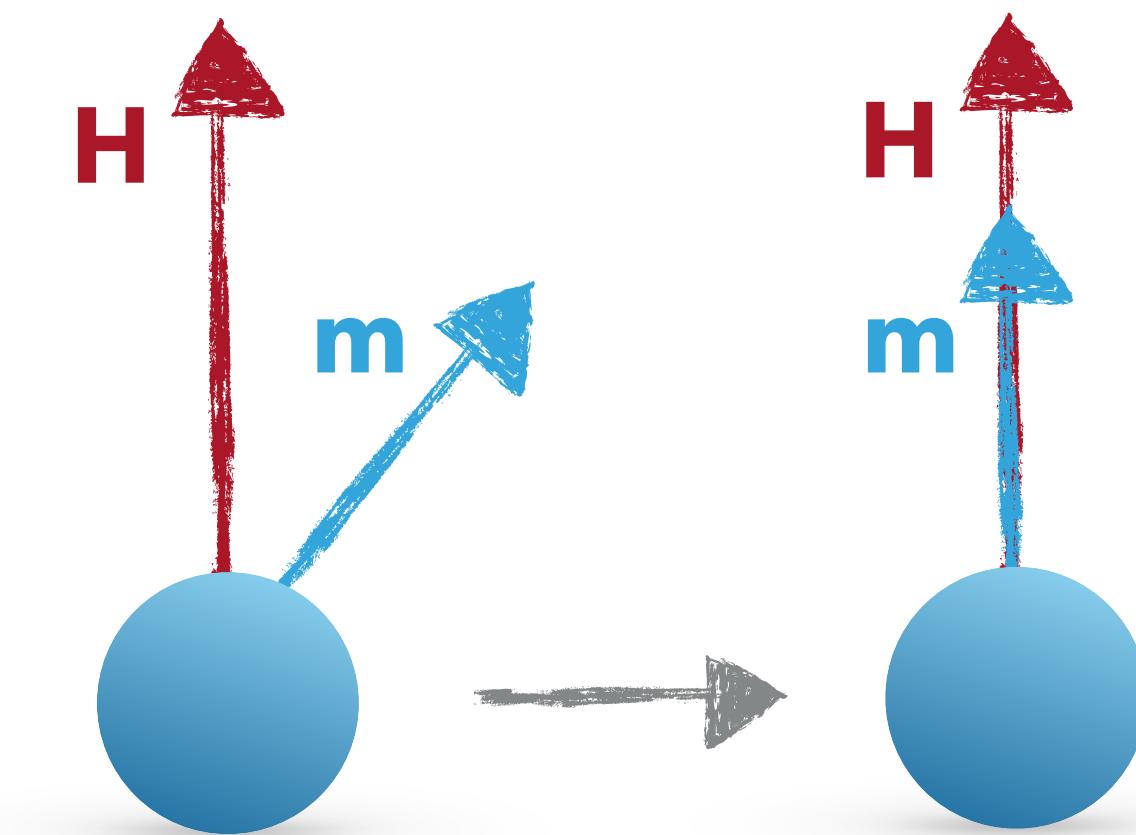


ZEEMAN

- ▶ Aligns \mathbf{m} parallel to \mathbf{H} .
- ▶ Parameter: \mathbf{H} (A/m)

Parameter in
ubermag is \mathbf{H}
and not \mathbf{B} .

$$\begin{aligned} w_z &= -\mathbf{M} \cdot \mathbf{B} & (\mathbf{B} = \mu_0 \mathbf{H}, \mathbf{M} = M_s \mathbf{m}) \\ &= -\mu_0 M_s \mathbf{m} \cdot \mathbf{H} \end{aligned}$$



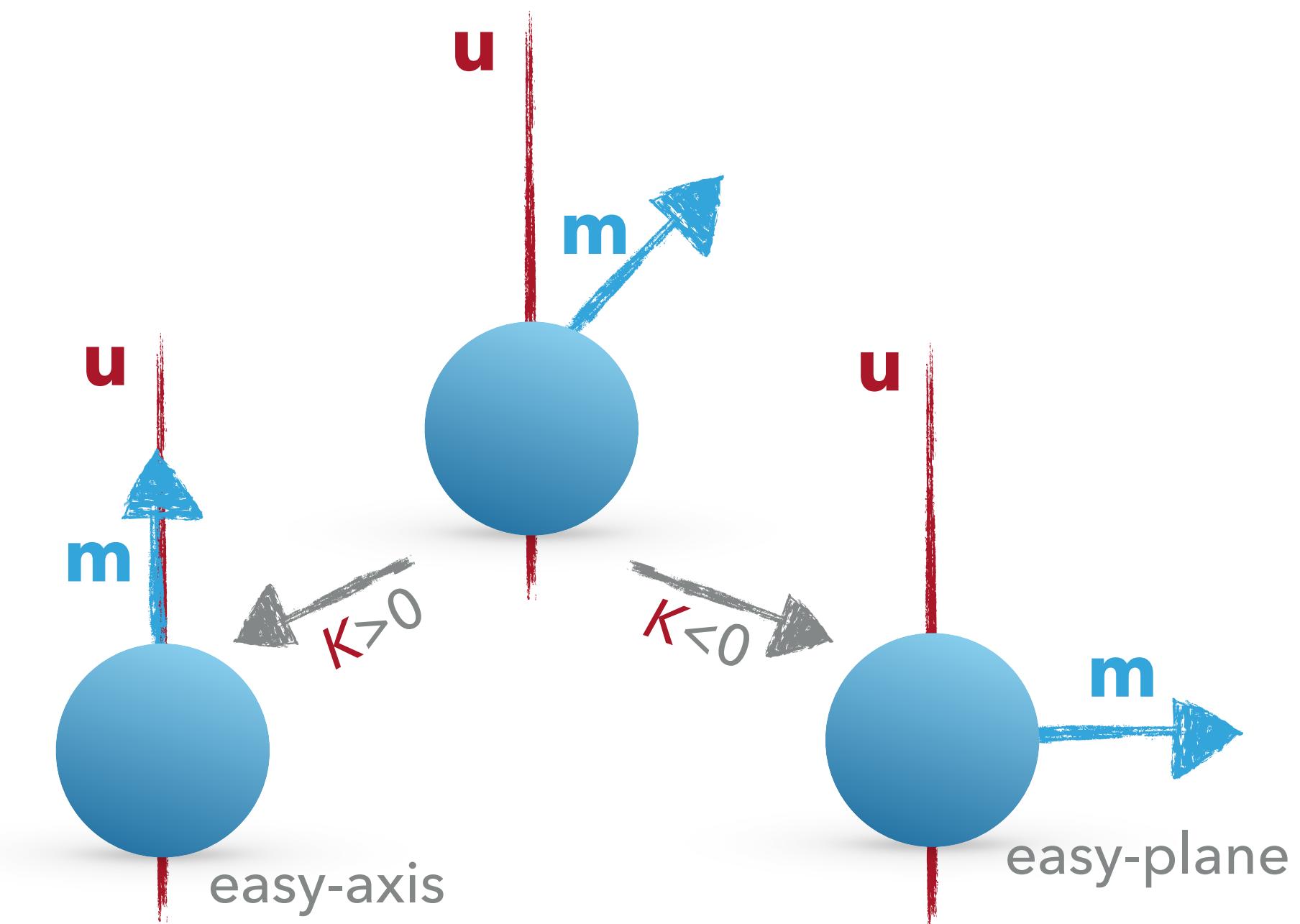
UNIAXIAL ANISOTROPY

- ▶ Aligns \mathbf{m} along \mathbf{u} or perpendicular to \mathbf{u} .
- ▶ Parameters: K (J/m³), \mathbf{u}

$$\begin{aligned} w_{ua} &= -K(\mathbf{m} \cdot \mathbf{u})^2 \quad (|\mathbf{m}| = 1, |\mathbf{u}| = 1) \\ &= -K \cos^2 \theta \\ &= -K(1 - \sin^2 \theta) = \cancel{-K} + K \sin^2 \theta \end{aligned}$$

const.

- ▶ There are several mathematically equivalent ways of writing uniaxial anisotropy.
- ▶ Changing on the sign of K , changes the “character” of uniaxial anisotropy.



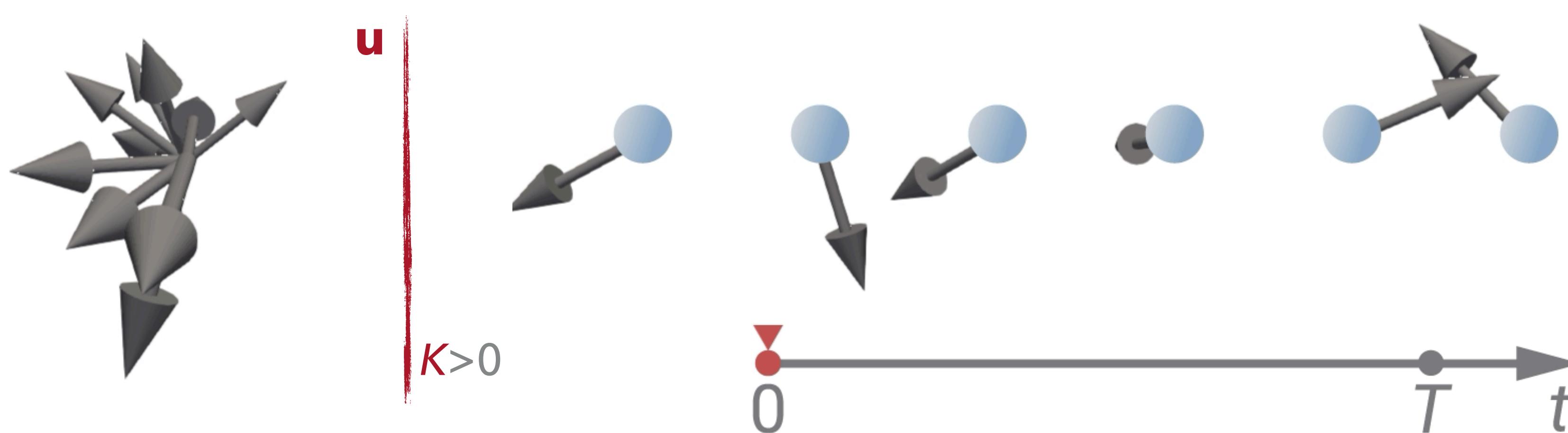
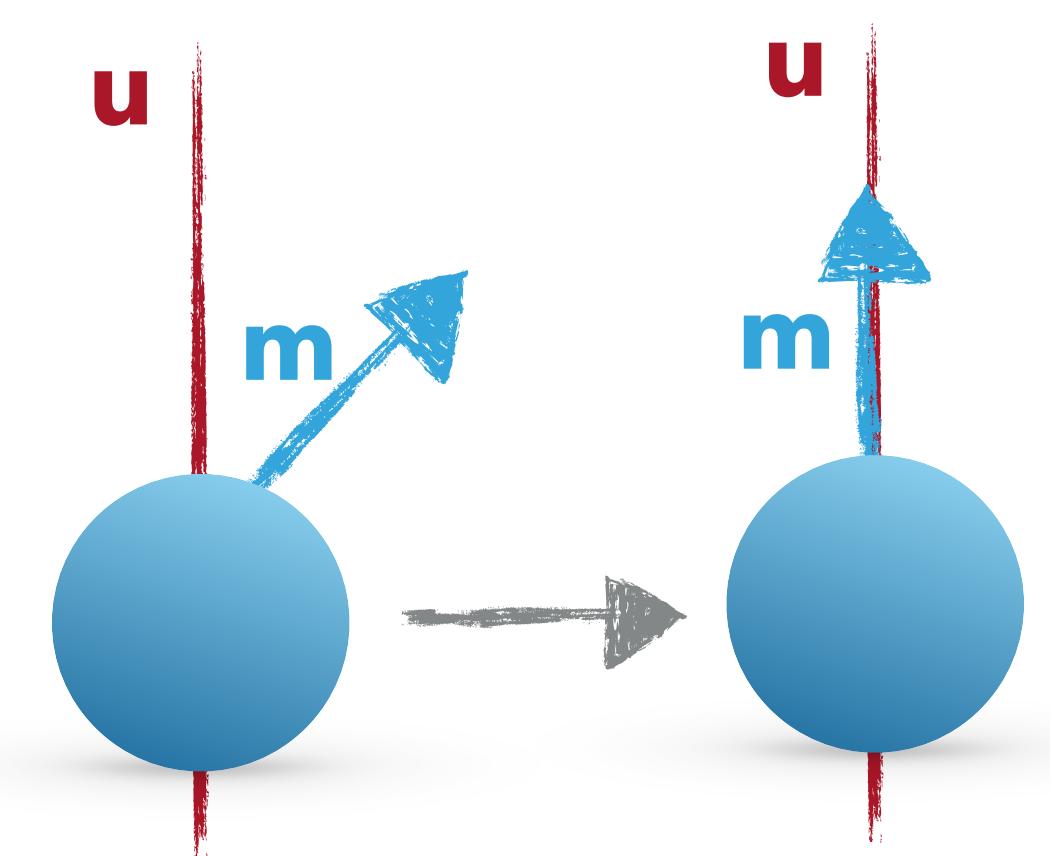
Be careful about
the sign of K
and if $|\mathbf{u}|=1$.

EASY AXIS UNIAXIAL ANISOTROPY

- Aligns \mathbf{m} parallel to antiparallel to \mathbf{u} .
- Parameters: $K > 0$ (J/m^3), \mathbf{u}

Be careful that
 $K > 0$ and $|\mathbf{u}| = 1$.

$$w_{\text{ua}} = -K(\mathbf{m} \cdot \mathbf{u})^2 \quad (|\mathbf{m}| = 1, |\mathbf{u}| = 1)$$

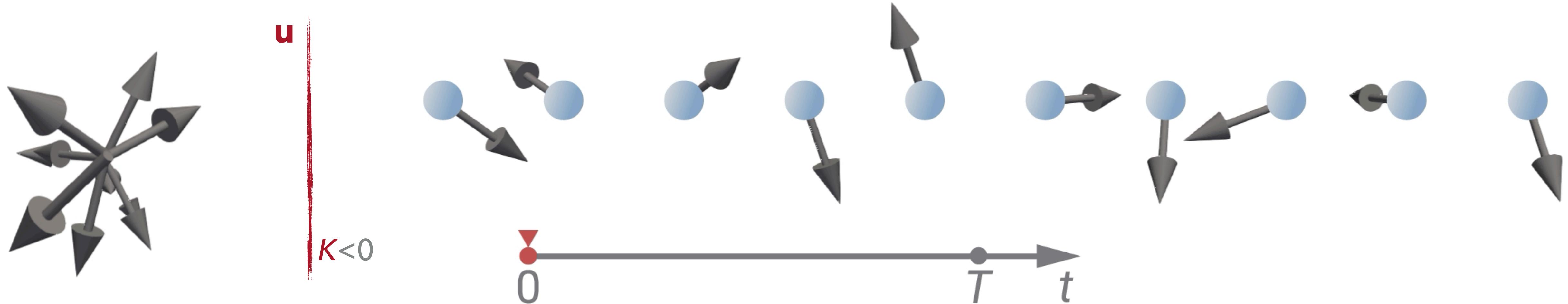
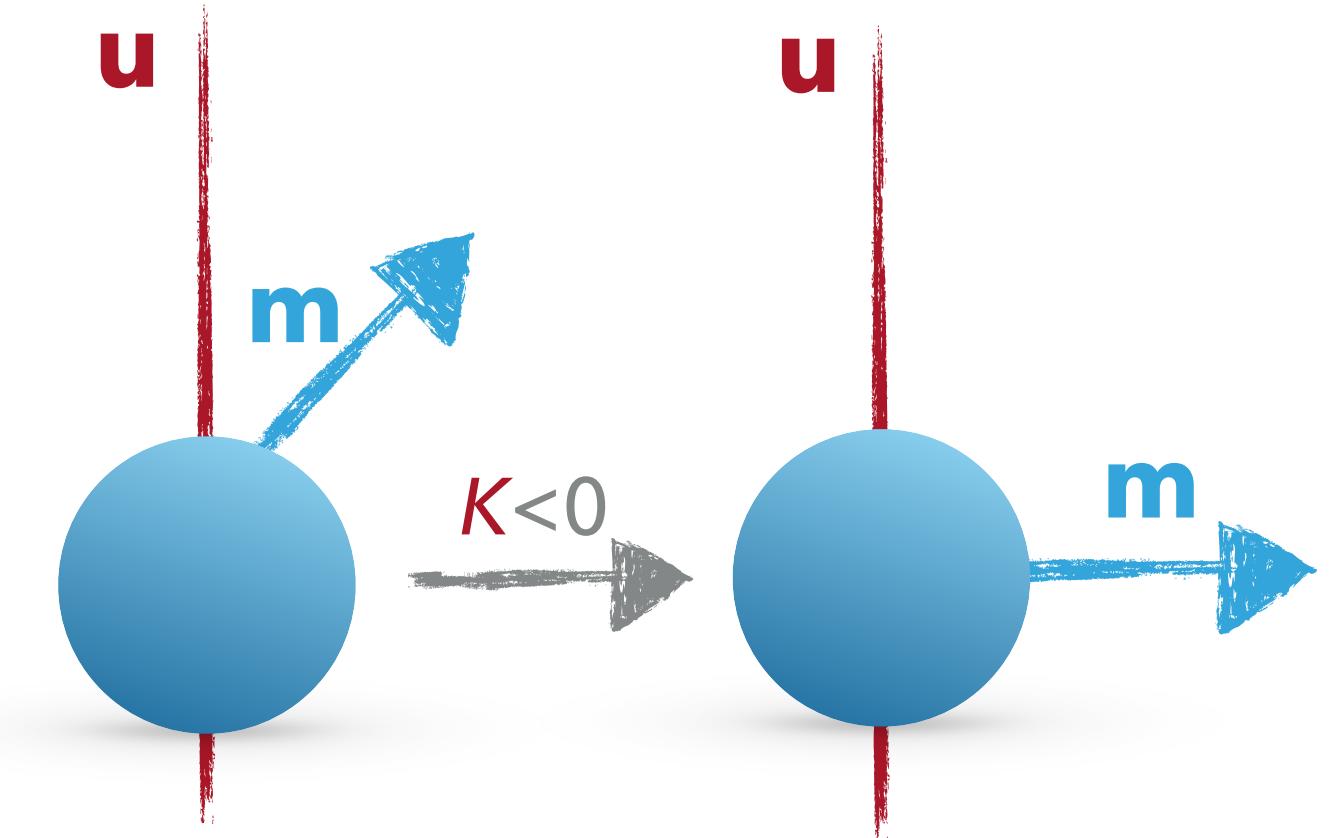


EASY PLANE UNIAXIAL ANISOTROPY

- ▶ Aligns \mathbf{m} perpendicular to \mathbf{u} .
- ▶ Parameters: $K < 0$ (J/m^3), \mathbf{u}

Be careful that
 $K < 0$ and $|\mathbf{u}| = 1$.

$$w_{ua} = -K(\mathbf{m} \cdot \mathbf{u})^2 \quad (|\mathbf{m}| = 1, |\mathbf{u}| = 1)$$



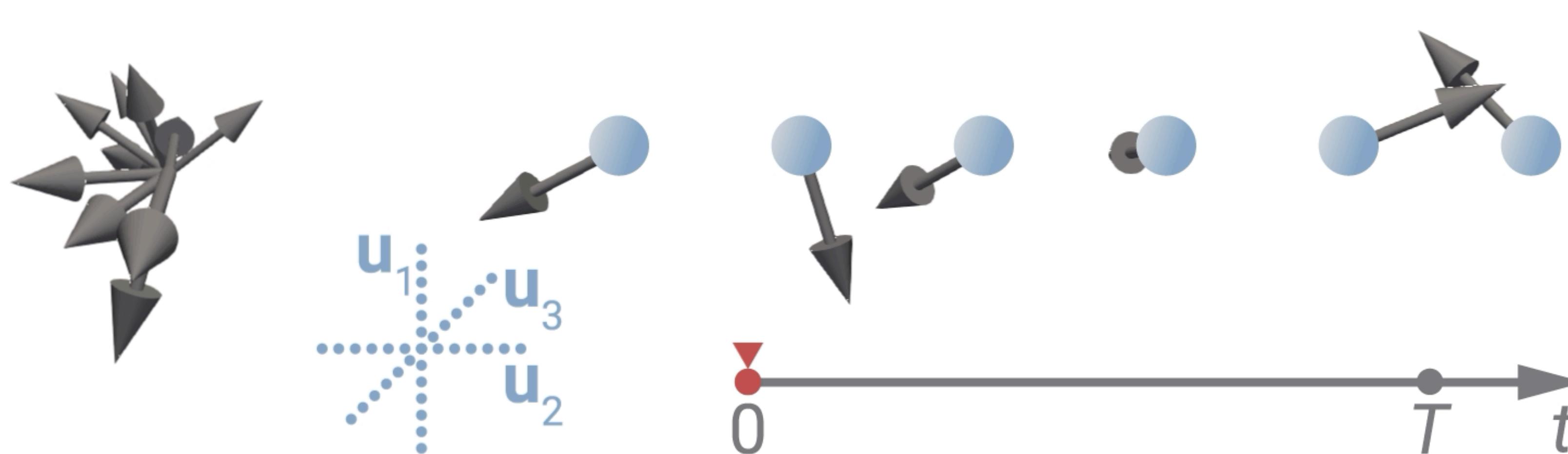
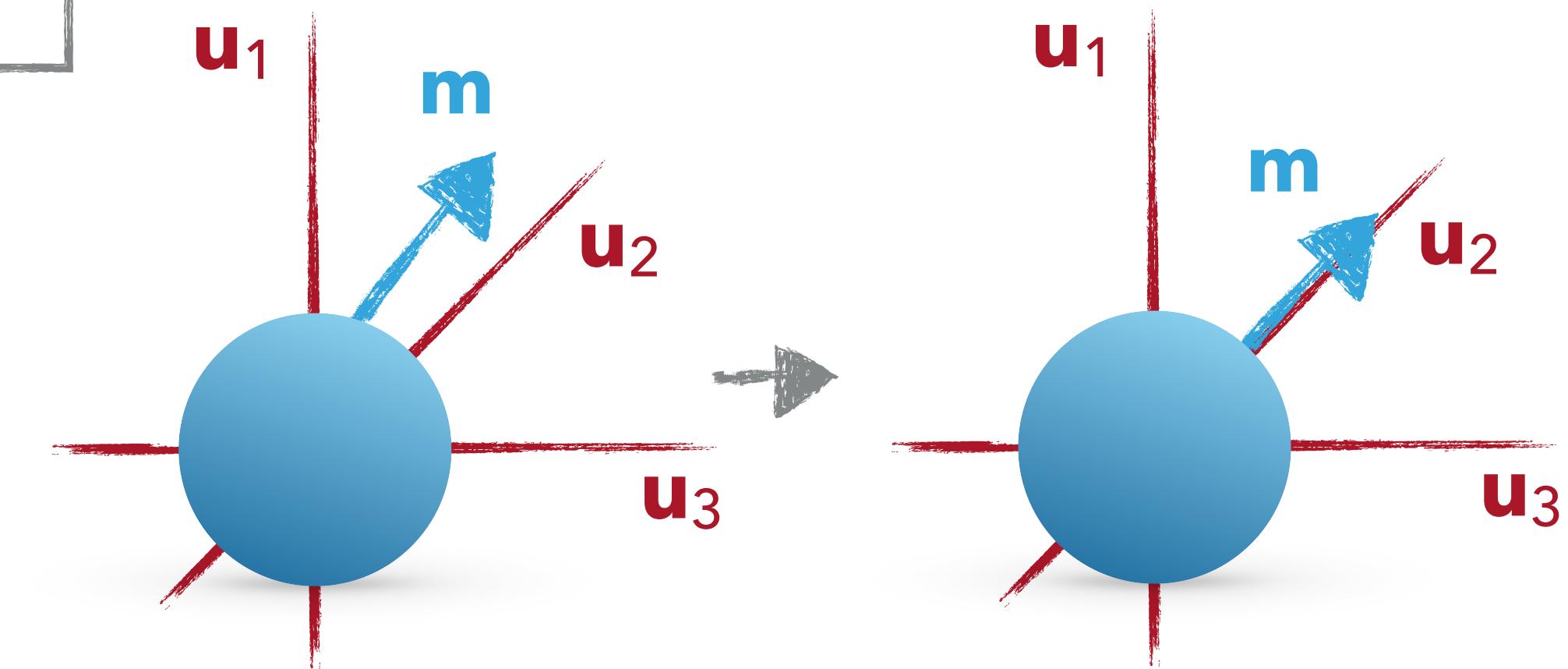
CUBIC ANISOTROPY

- Aligns \mathbf{m} along \mathbf{u}_1 , \mathbf{u}_2 , or \mathbf{u}_3 .
- Parameters: K (J/m³), \mathbf{u}_1 , \mathbf{u}_2 , (and) \mathbf{u}_3

$$w_{ca} = -K(m_1^2 m_2^2 + m_2^2 m_3^2 + m_1^2 m_3^2)$$

$$(|\mathbf{m}| = 1, |\mathbf{u}| = 1, m_i = \mathbf{m} \cdot \mathbf{u}_i)$$

Be careful if $|\mathbf{u}_i|=1$
and $\mathbf{u}_3 = \mathbf{u}_1 \times \mathbf{u}_2$



(FERROMAGNETIC) EXCHANGE

- Aligns all magnetic moments (in \mathbf{m}) parallel to each other.
- Parameter: A (J/m)

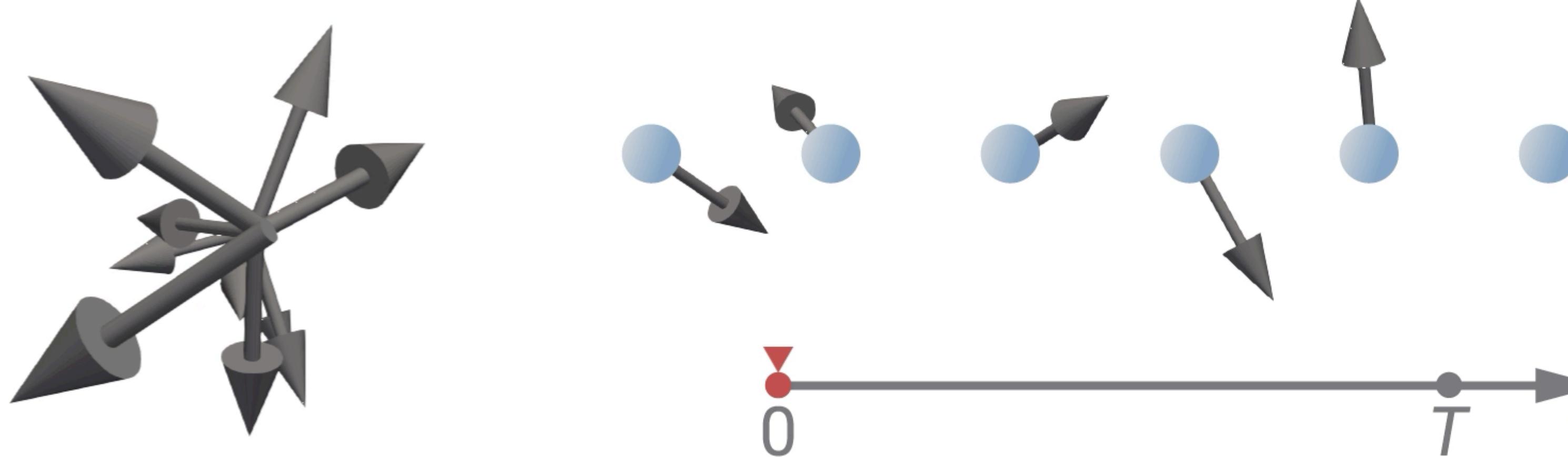
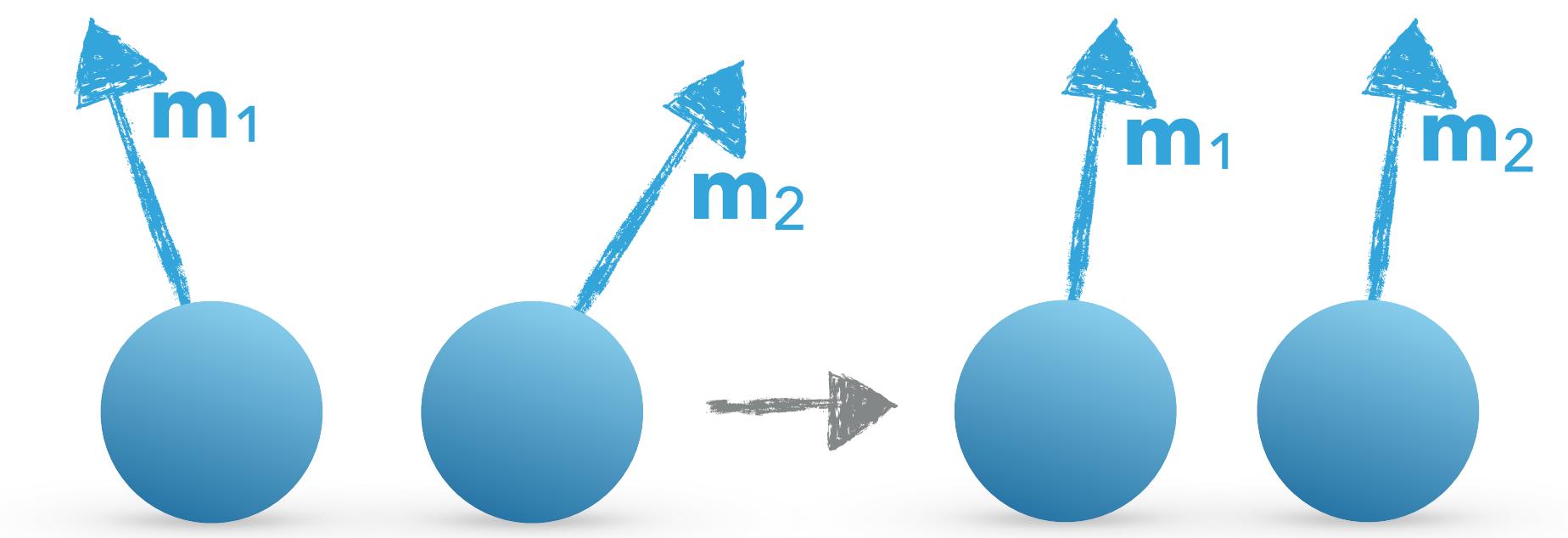
$$w_{\text{ex}} = A \mathbf{m} \cdot \nabla^2 \mathbf{m} \quad (\nabla^2 \mathbf{m} = \nabla^2 m_x \hat{\mathbf{x}} + \nabla^2 m_y \hat{\mathbf{y}} + \nabla^2 m_z \hat{\mathbf{z}})$$

$$= A[(\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2]$$

$$(= A(\nabla \mathbf{m})^2)$$

vector Laplacian
just a convention

Using $A < 0$ (antiferromagnetic exchange) in micromagnetics is not justified.

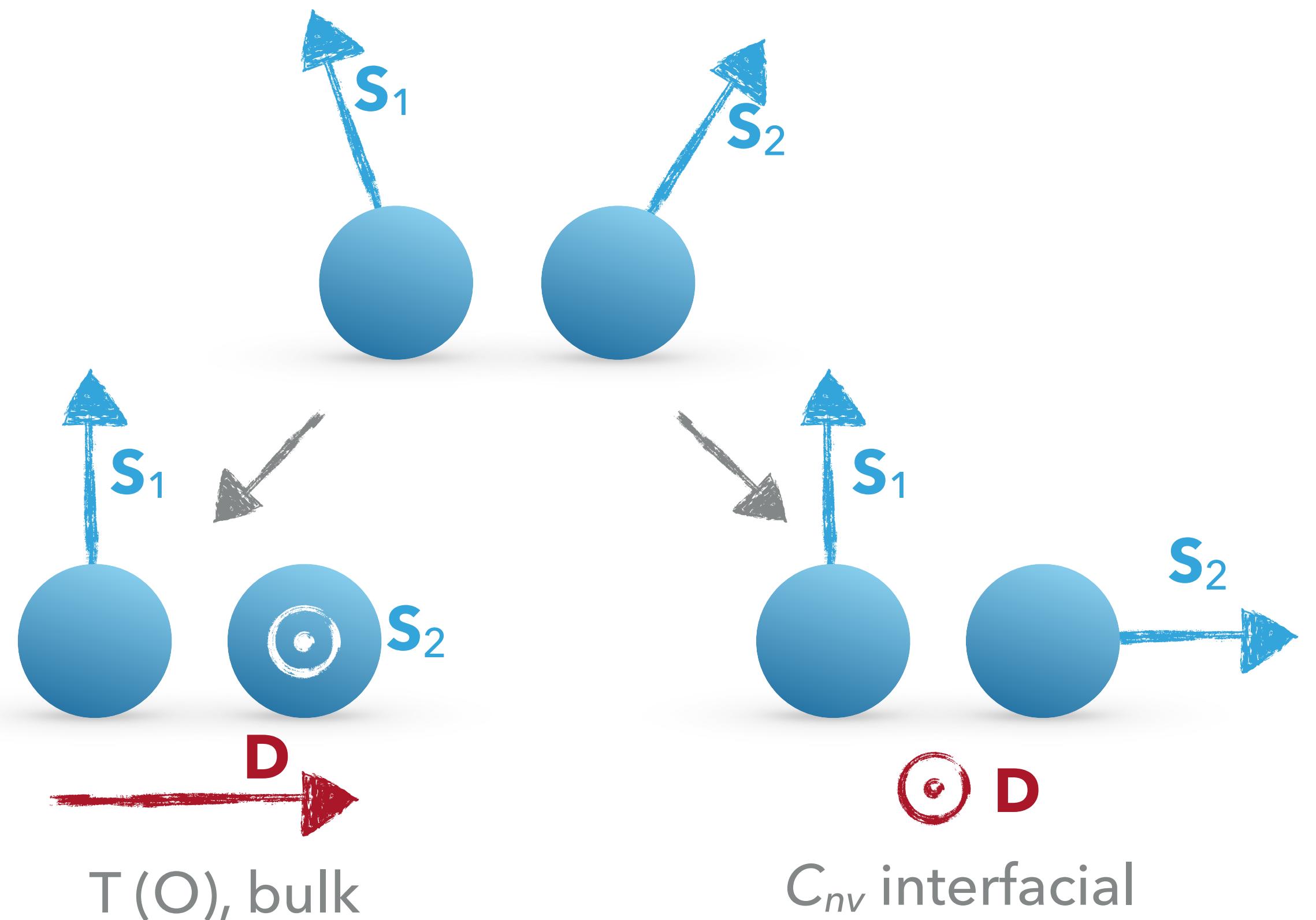


(ATOMISTIC) DZYALOSHINSKII-MORIYA

- Aligns neighbouring magnetic moments (in \mathbf{m}) perpendicular to each other and perpendicular to \mathbf{D} .

$$w_{ij}^{\text{dmi}} = (\pm) \mathbf{D} \cdot (\mathbf{S}_i \times \mathbf{S}_j)$$

Be careful about the sign.
There is no clear convention.

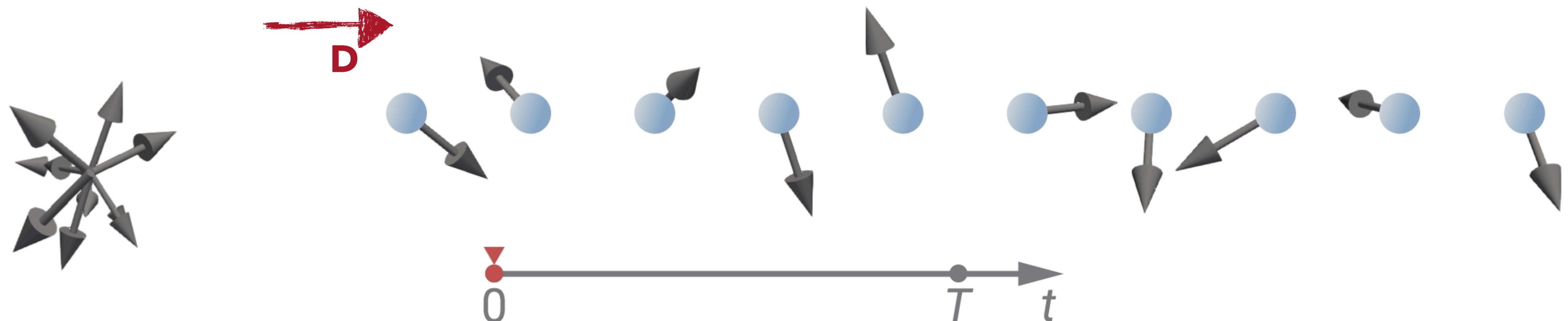
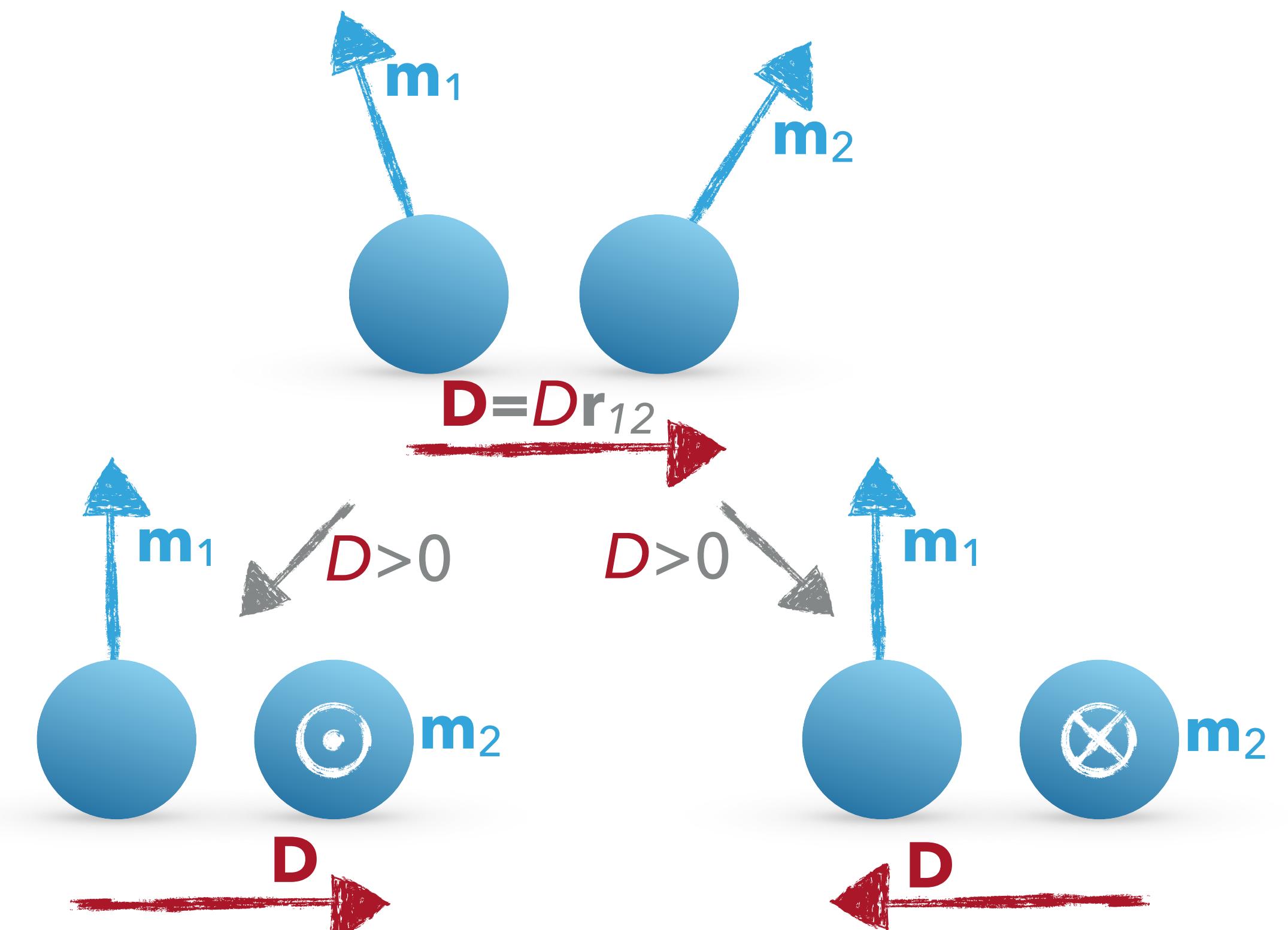


DZYALOSHINSKII-MORIYA (T, 0)

- Aligns neighbouring magnetic moments (in \mathbf{m}) perpendicular to each other.
- Parameter: D (J/m^2)

$$\omega_{\text{dmi}} = (\pm) D \mathbf{m} \cdot (\nabla \times \mathbf{m}) \quad (D = D \mathbf{r}_{ij})$$

Be careful about the sign of D .
There is no clear convention.



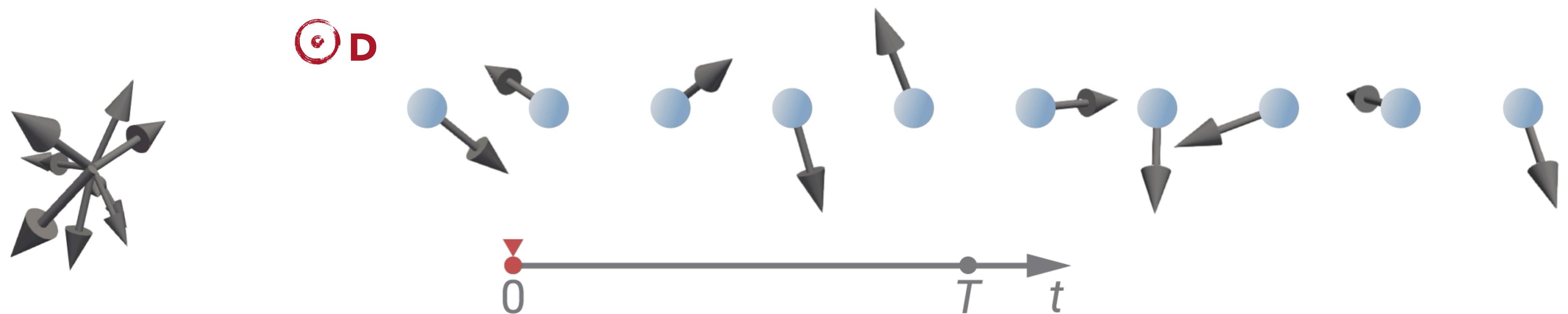
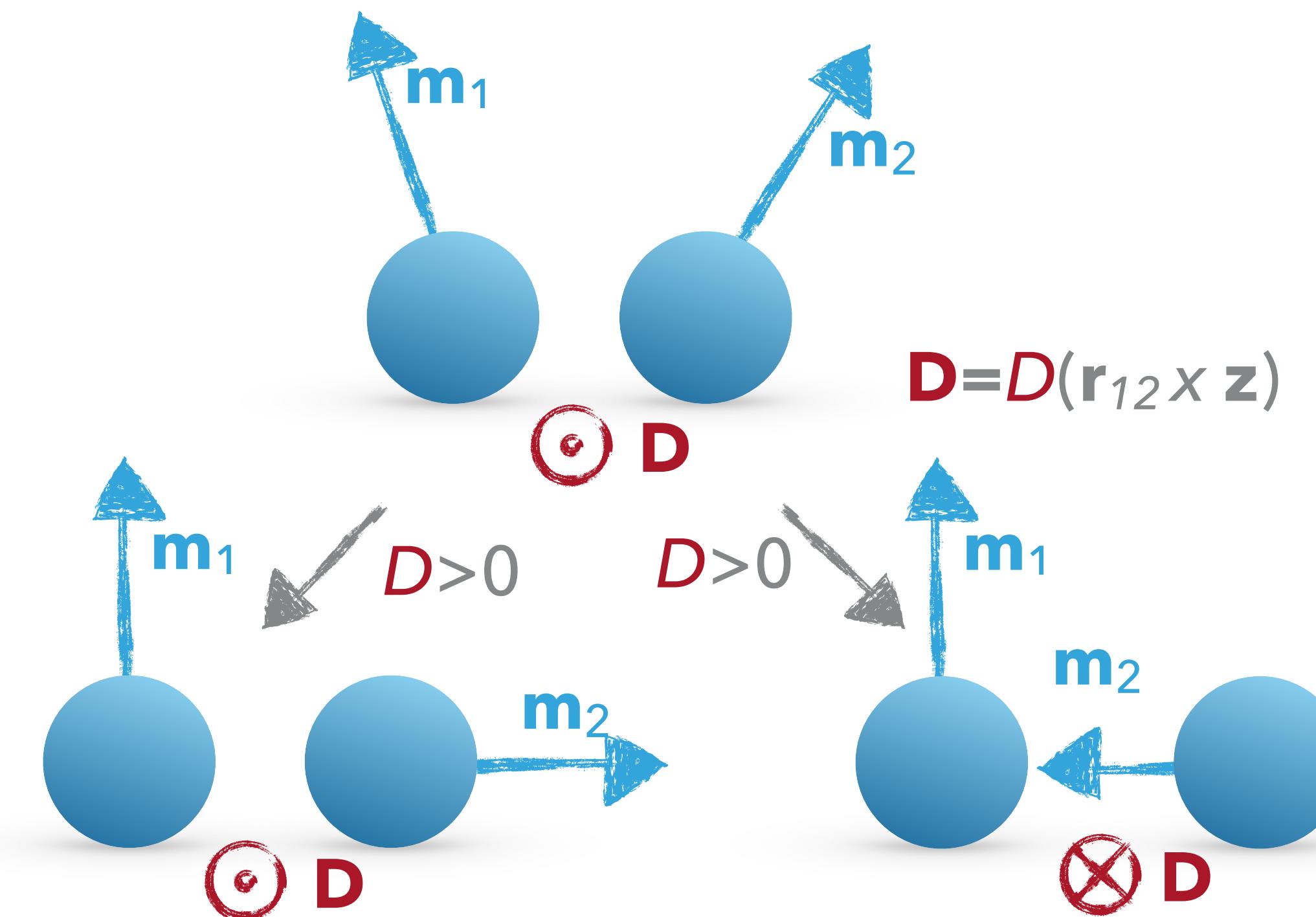
DZYALOSHINSKII-MORIYA (C_{NV})

- Aligns neighbouring magnetic moments (in \mathbf{m}) perpendicular to each other.
- Parameter: D (J/m²)

$$\omega_{\text{dmi}} = (\pm)D(\mathbf{m} \cdot \nabla m_z - m_z \nabla \cdot \mathbf{m})$$

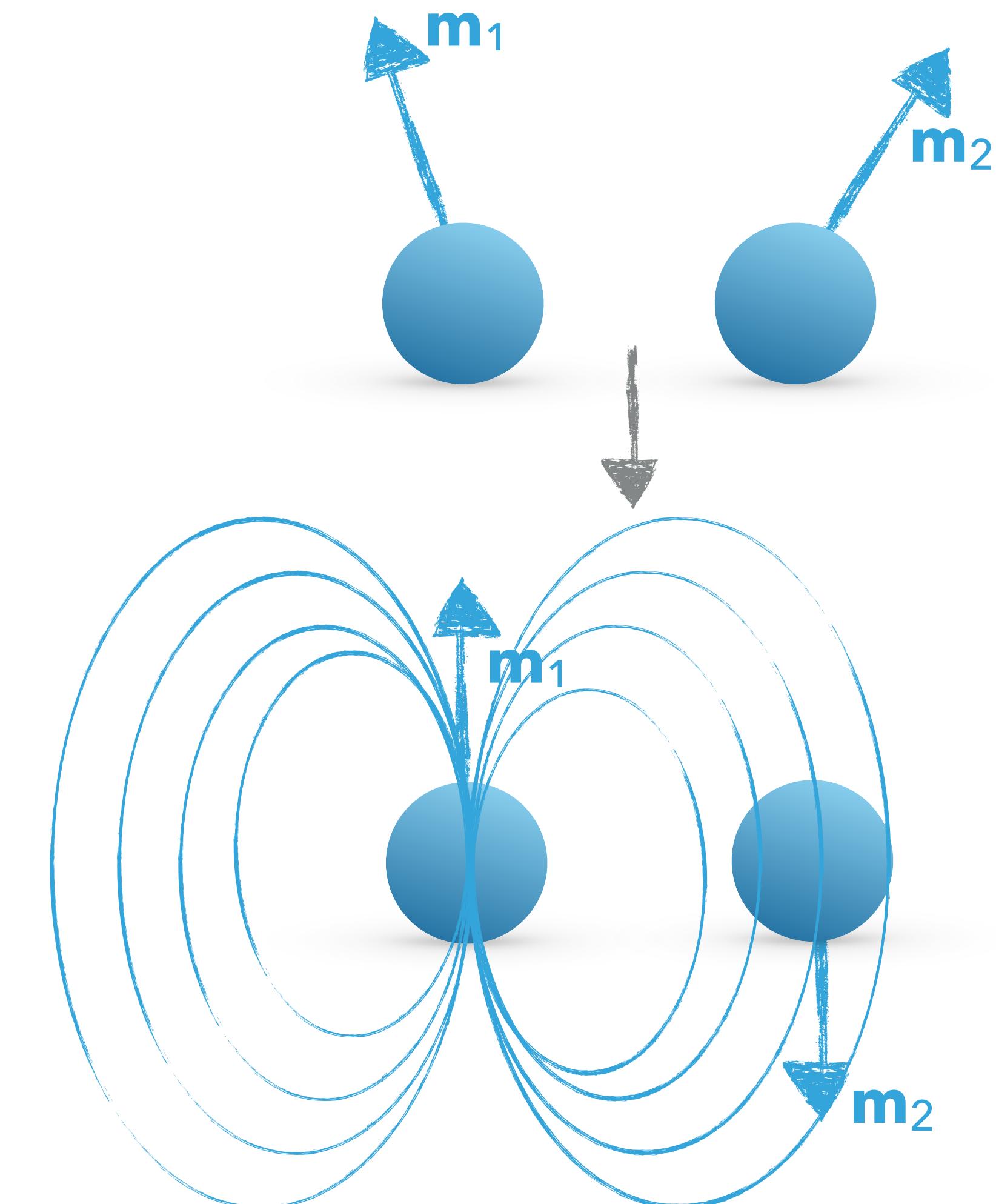
Be careful about the sign of D .
 There is no clear convention.

$$(D = D(r_{ij} \times \hat{\mathbf{z}}))$$

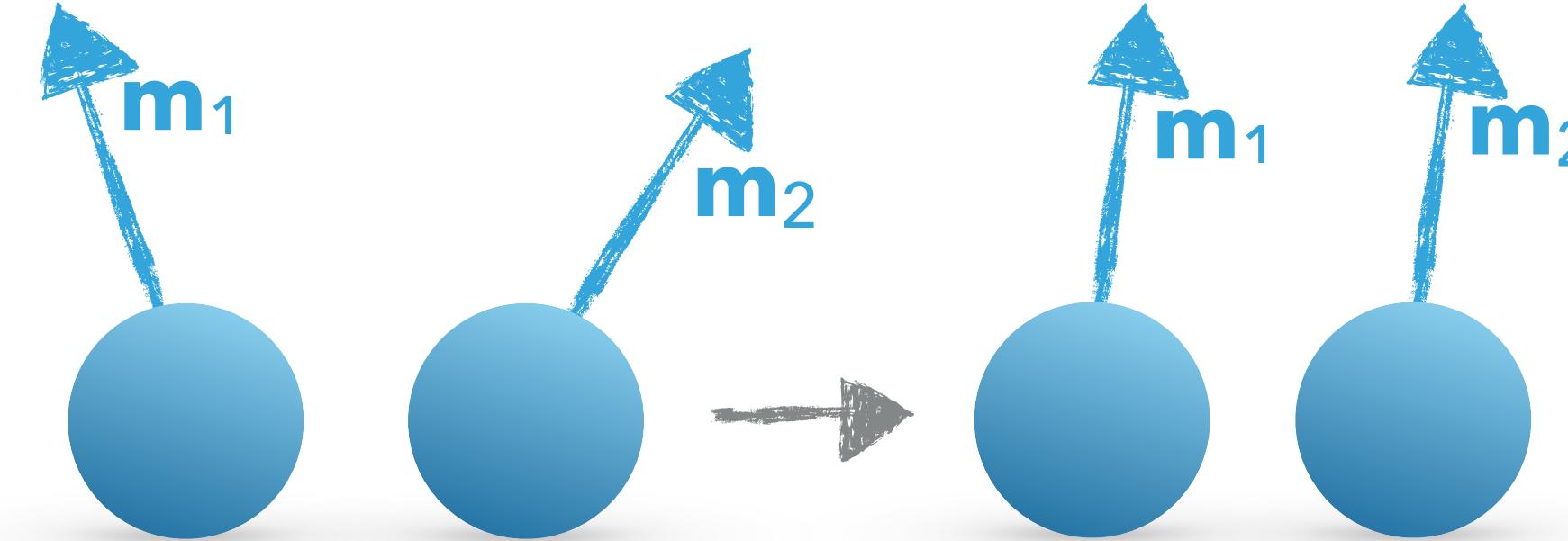


DEMAGNETISATION

- ▶ **Long range interaction**
- ▶ It is computed from \mathbf{m} and no parameter is required.
- ▶ Each magnetic moment (in \mathbf{m}) "feels" the total magnetic field of all surrounding moments.
- ▶ **Computationally expensive.**
- ▶ More details in the next sessions.

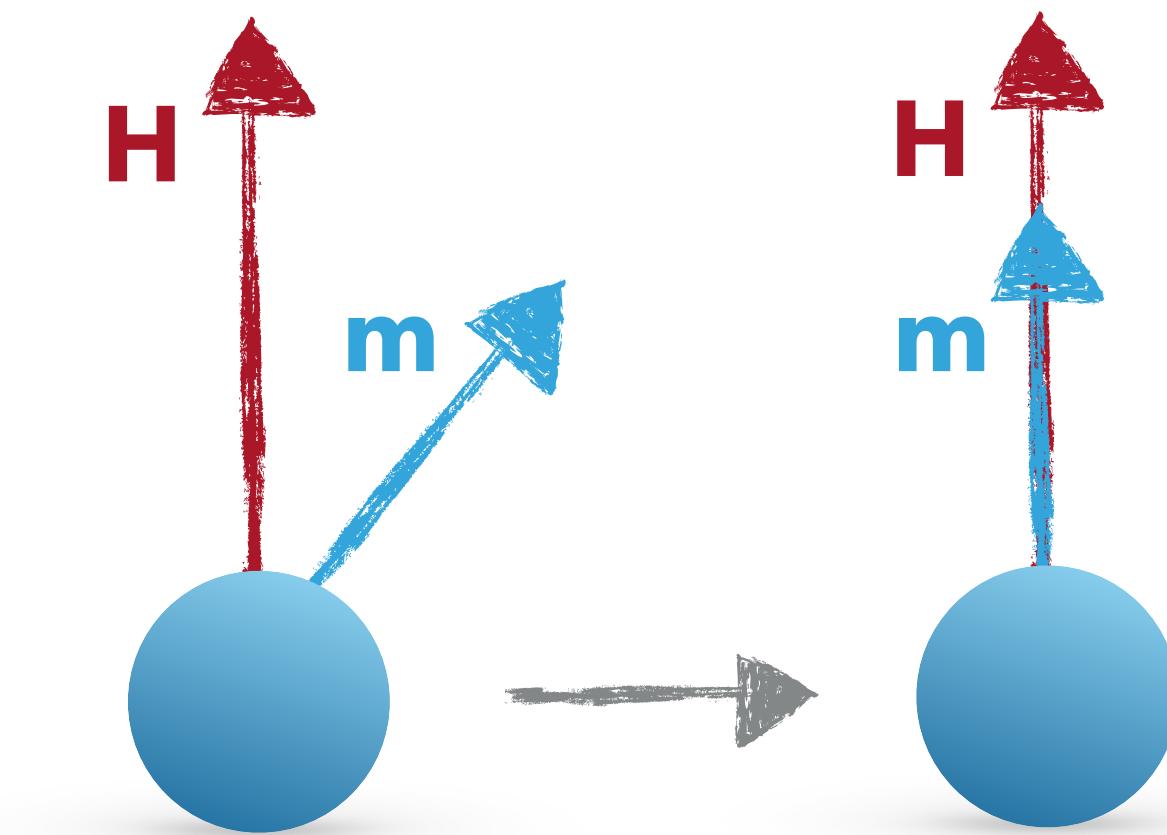


EXCHANGE AND ZEEMAN

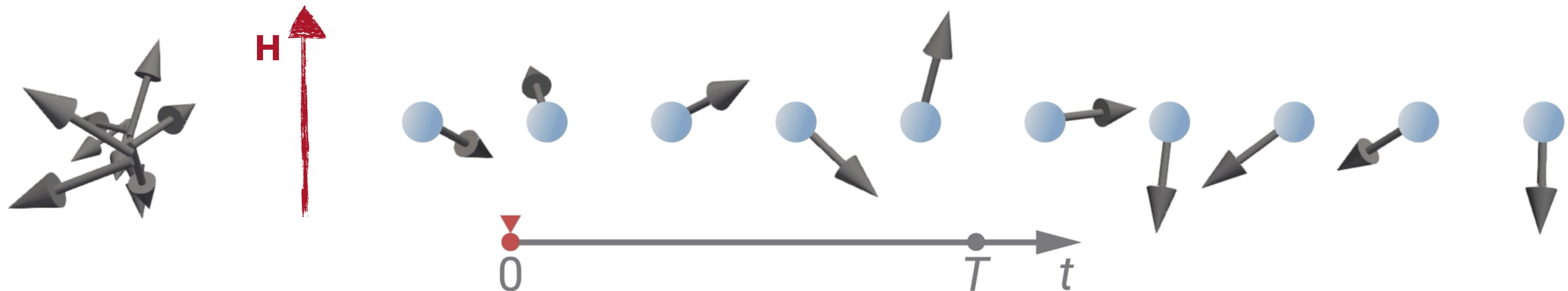


All magnetic moments parallel to each other?

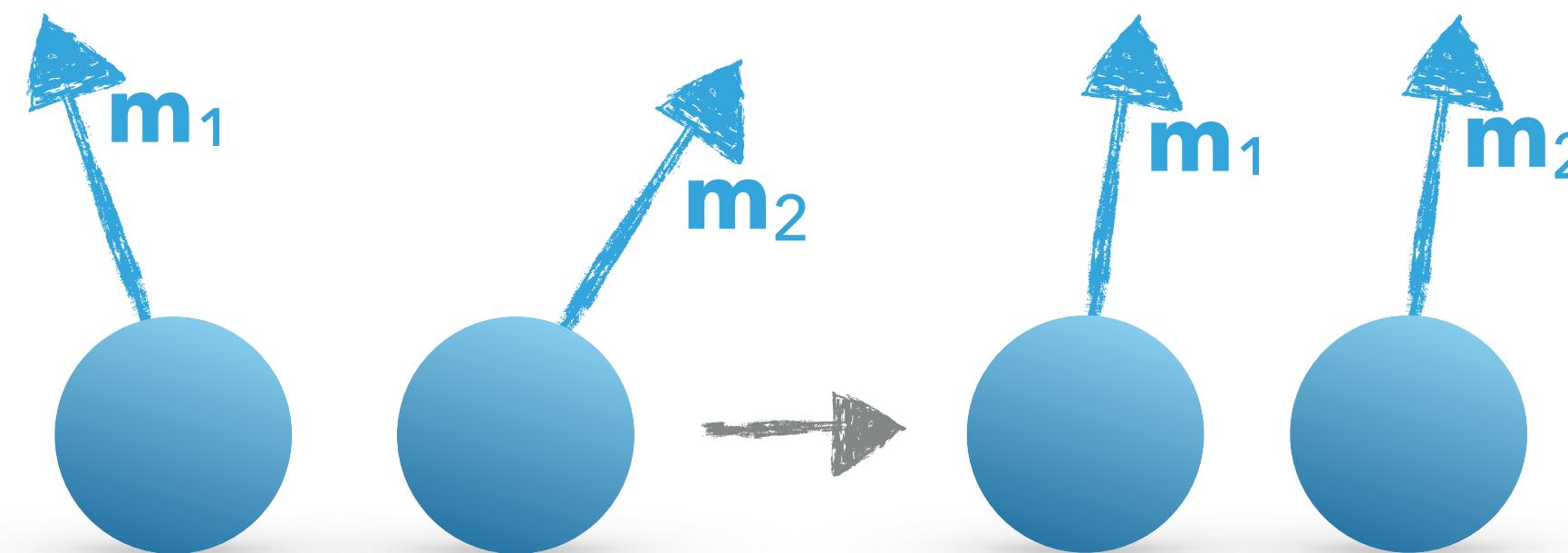
No compromise is needed.



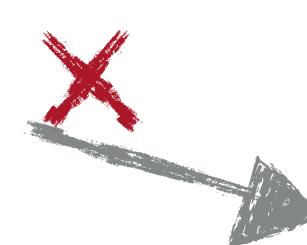
All magnetic moments parallel to H ?



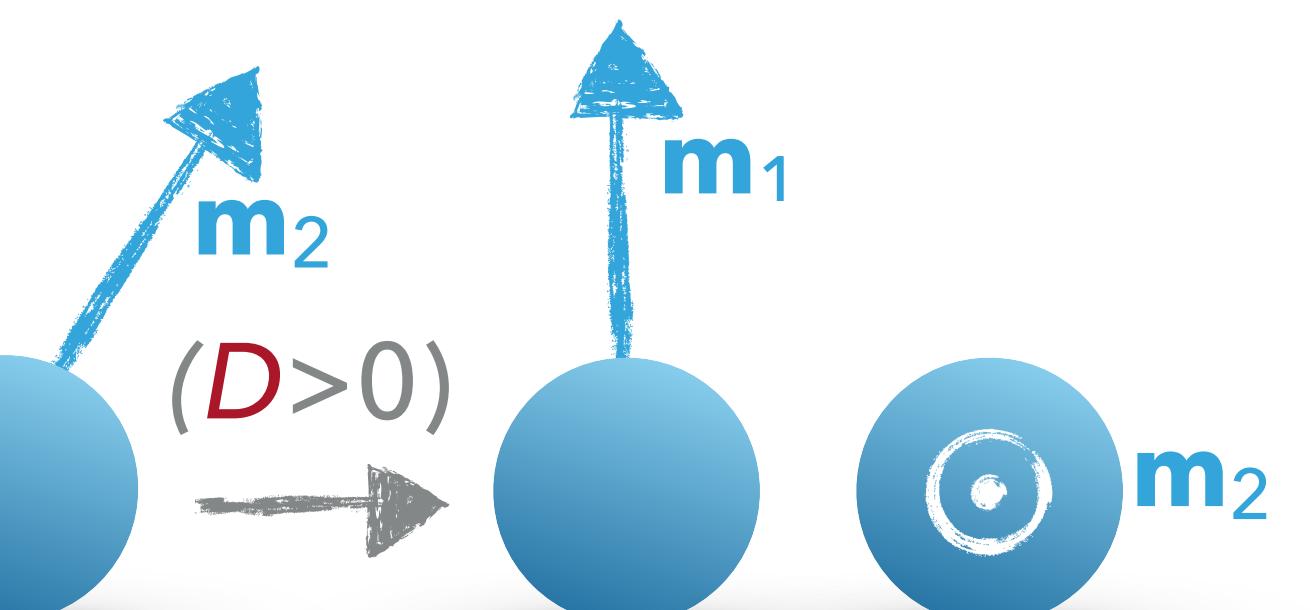
EXCHANGE AND DMI (T)



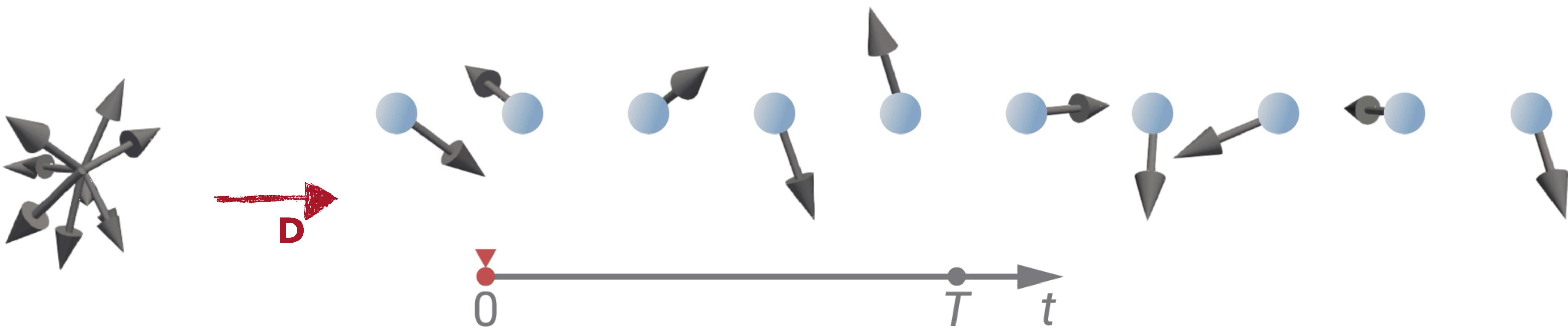
All magnetic moments
parallel to each other?



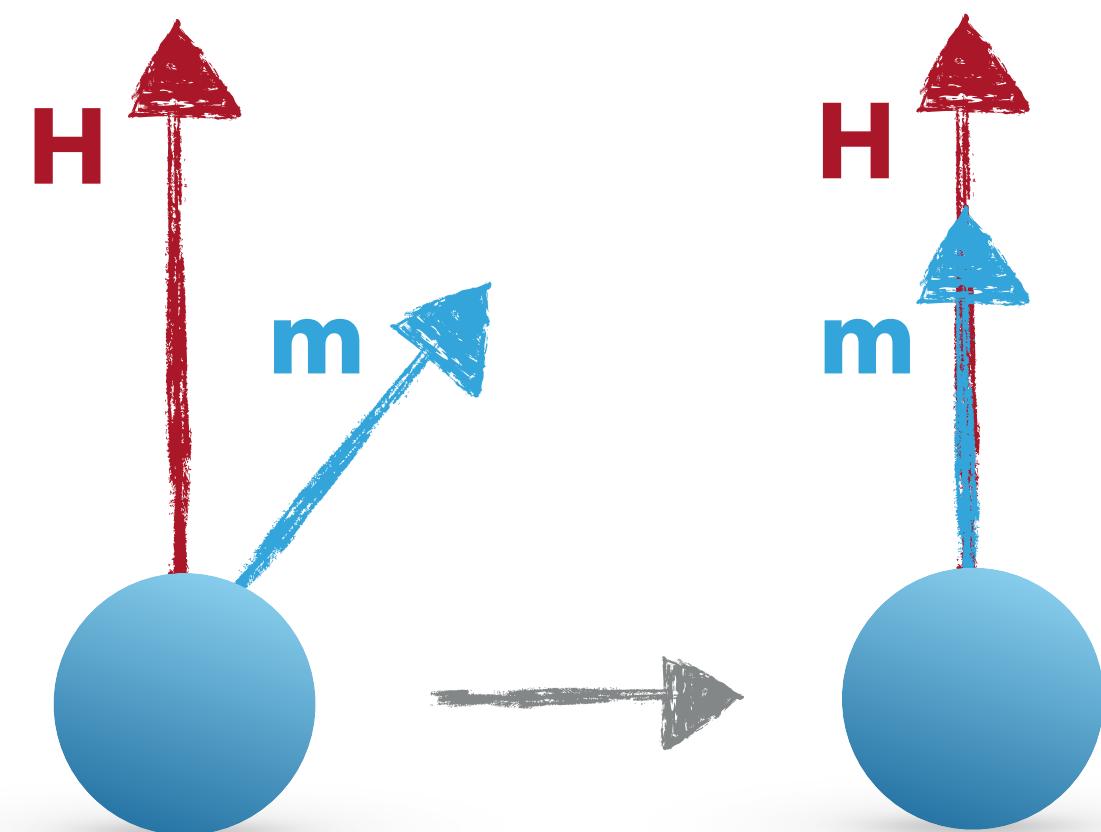
Energies have to
compete and reach
a compromise.



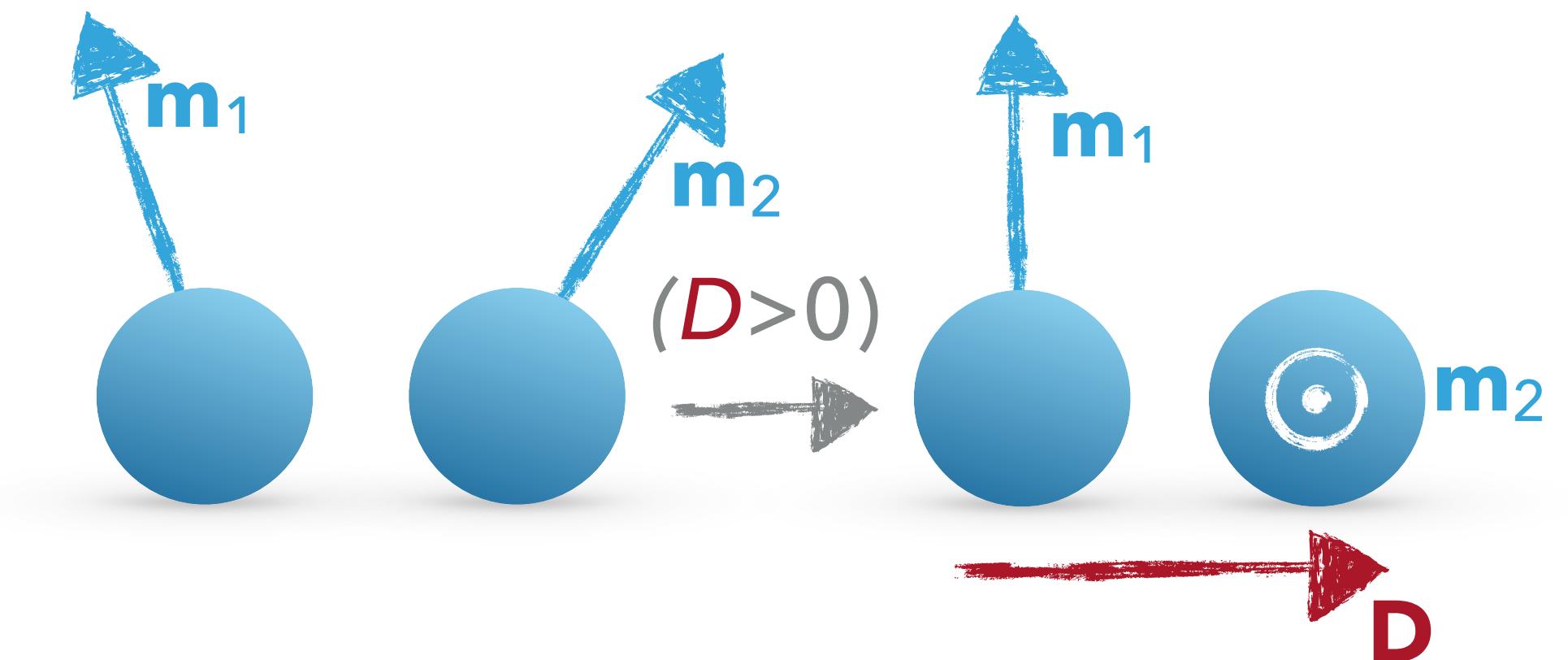
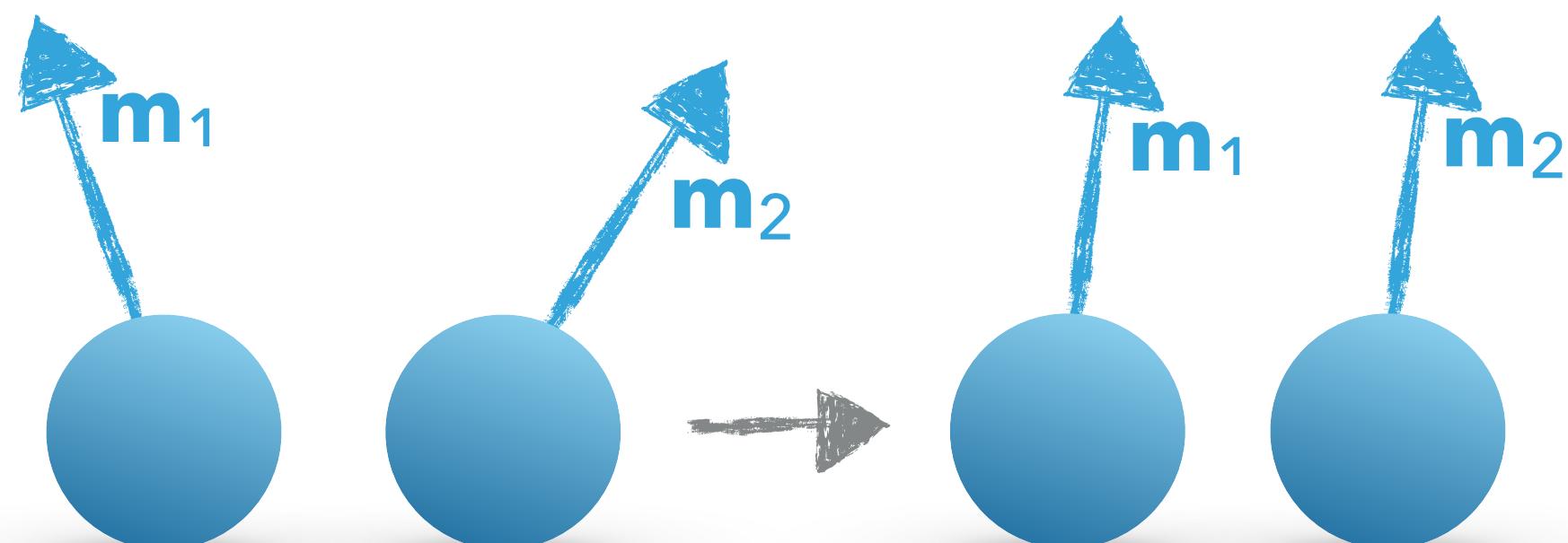
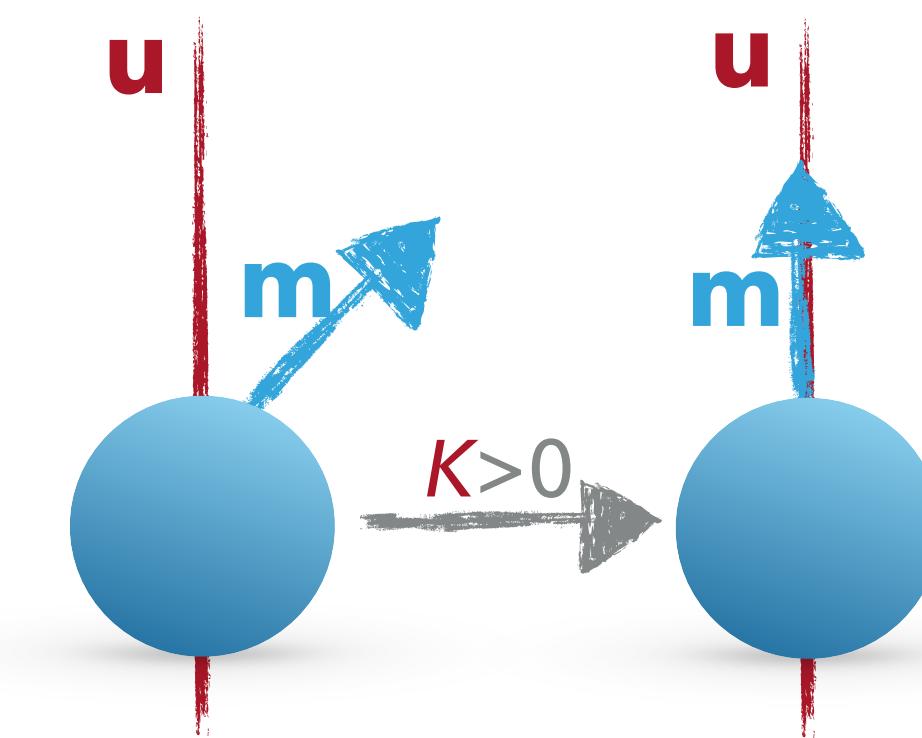
All magnetic moments
perpendicular to each
other?



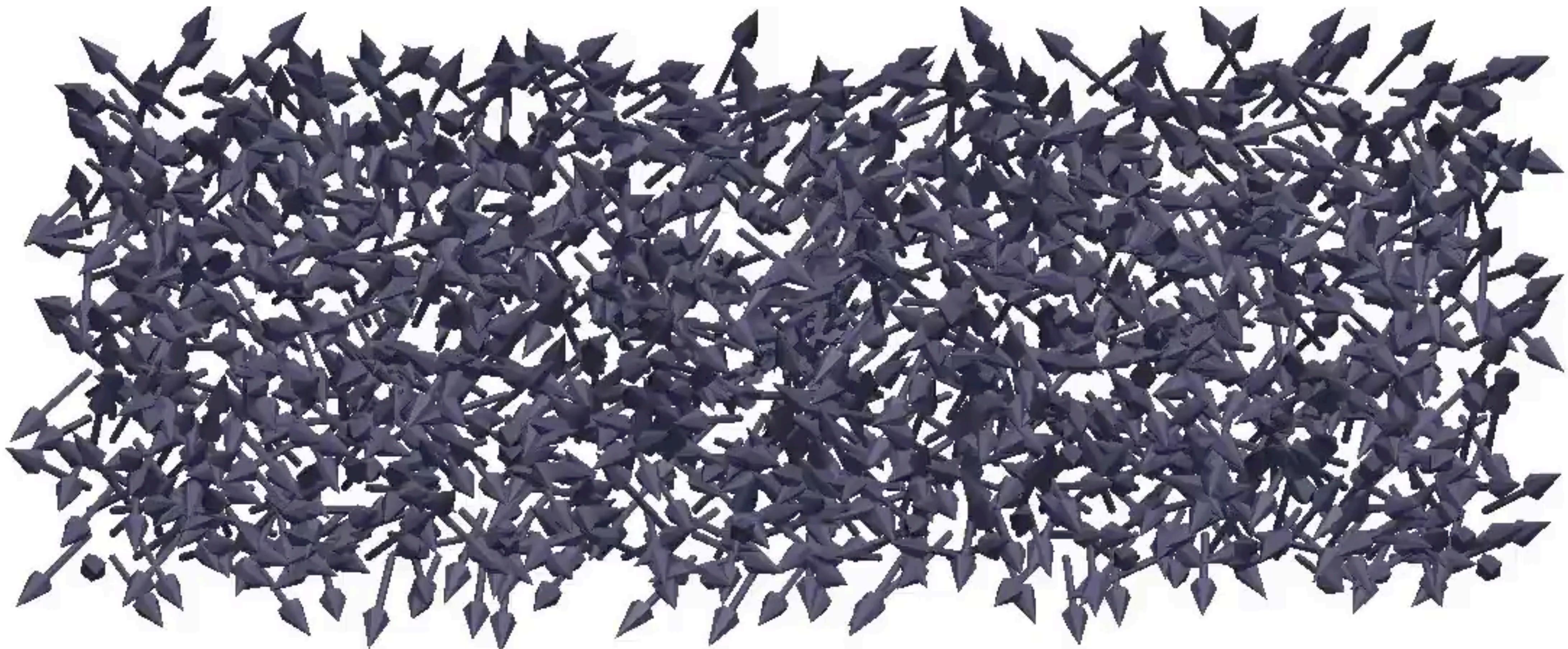
MORE COMPLICATED CASE IN A 2D SAMPLE (1/2)



Energies have to compete and reach a compromise.



MORE COMPLICATED CASE IN A 2D SAMPLE (2/2)



DYNAMICS EQUATION

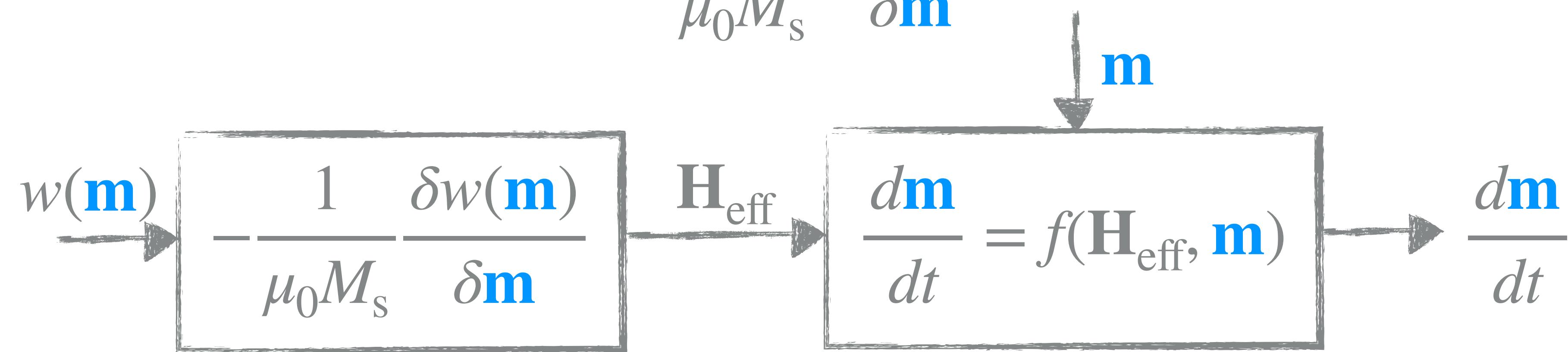
- ...tells us how magnetisation \mathbf{m} wants to change in order to minimise its energy.

$$\frac{d\mathbf{m}}{dt} = f_1(\mathbf{m}, \mathbf{H}_{\text{eff}}, \dots) + f_2(\mathbf{m}, \mathbf{H}_{\text{eff}}, \dots) + \dots = \sum_i f_i(\mathbf{m}, \mathbf{H}_{\text{eff}}, \dots)$$

user-defined

- Effective field is computed as the first variational derivative of energy density:

$$\mathbf{H}_{\text{eff}} = -\frac{1}{\mu_0 M_s} \frac{\delta w(\mathbf{m})}{\delta \mathbf{m}}$$

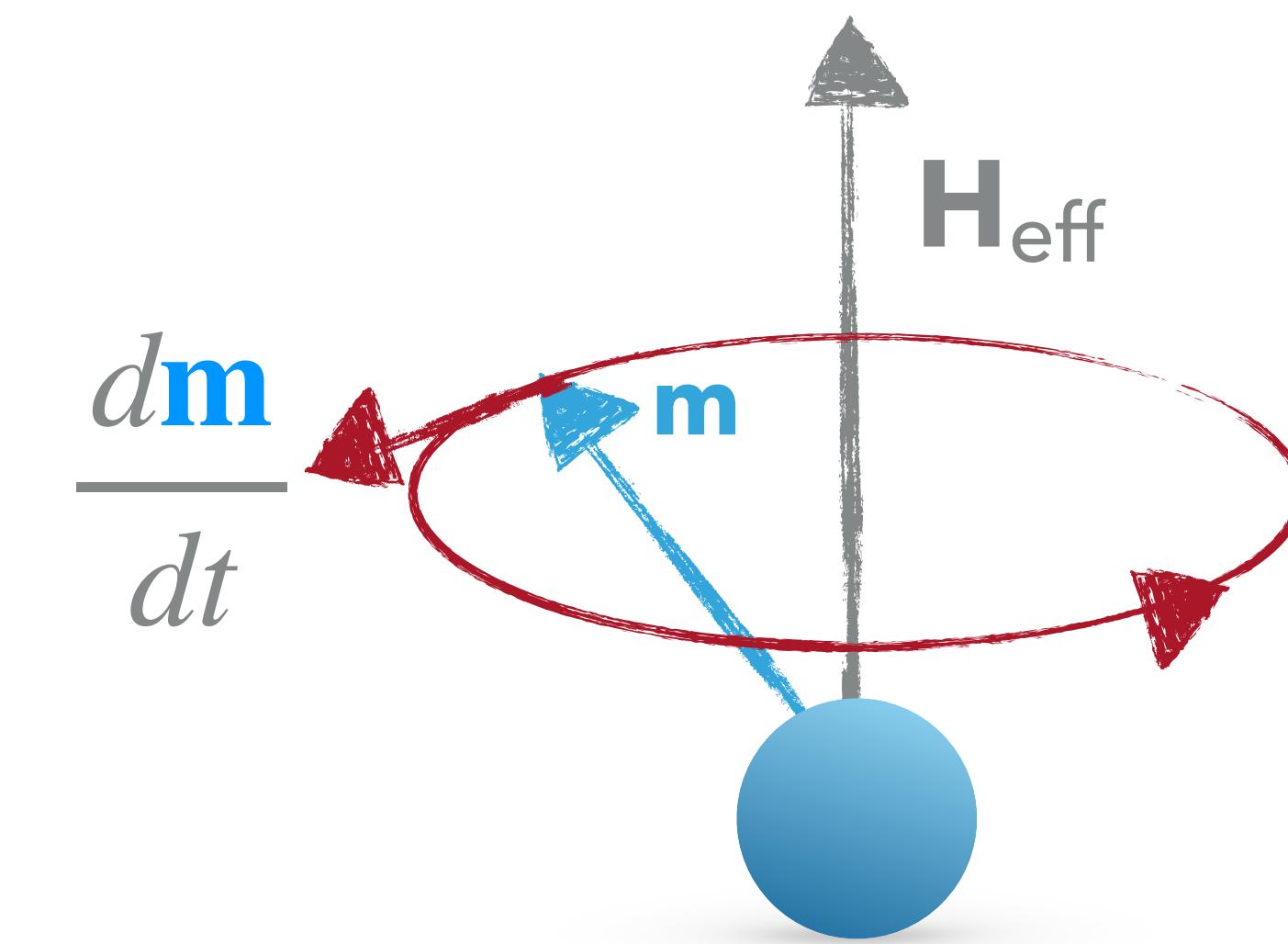
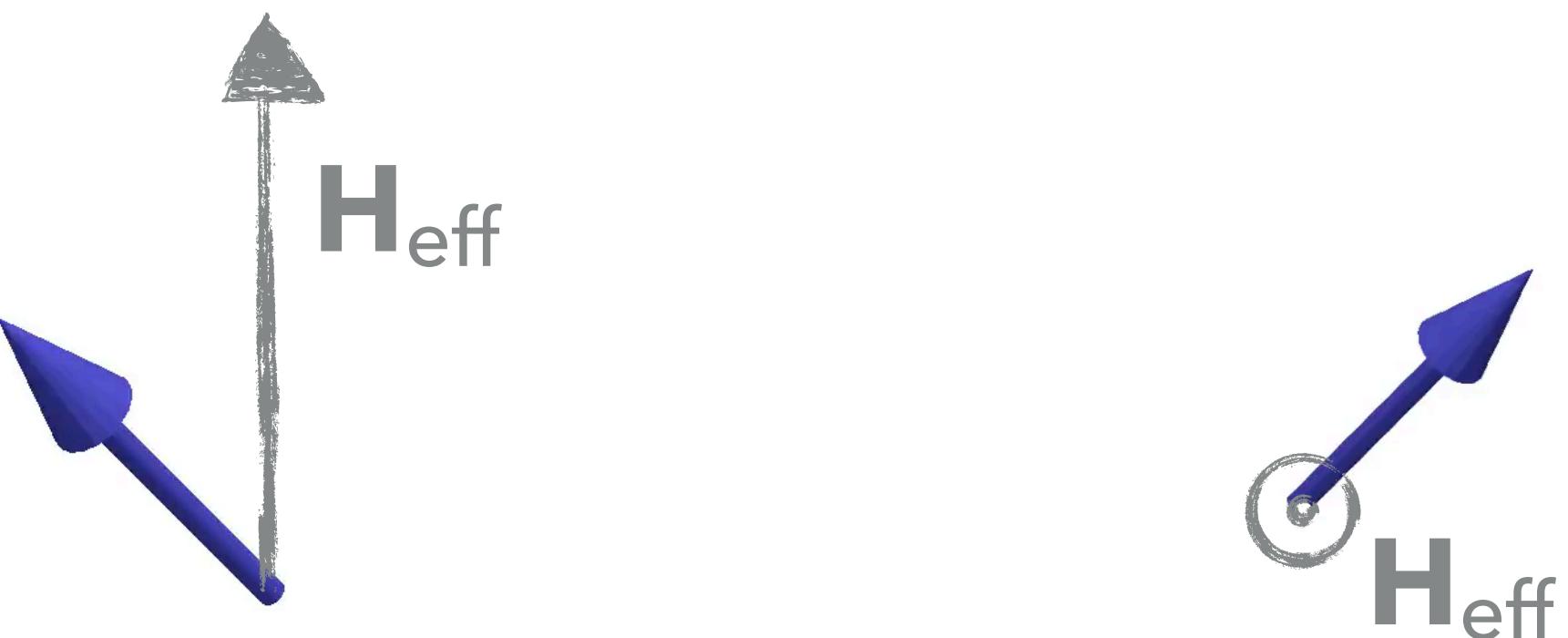


PRECESSION

- ▶ Makes \mathbf{m} precess around \mathbf{H}_{eff} .
- ▶ Parameter: γ_0

$$\frac{d\mathbf{m}}{dt} = -\gamma_0 \mathbf{m} \times \mathbf{H}_{\text{eff}} \quad (\gamma_0 = \mu_0 \gamma)$$

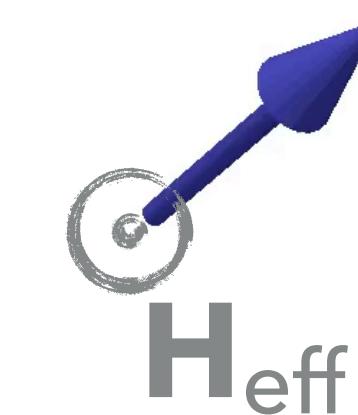
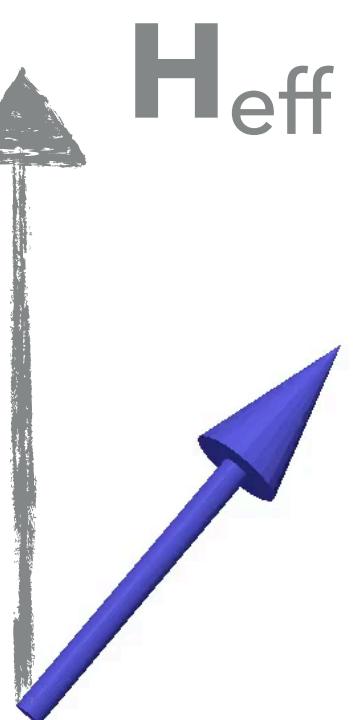
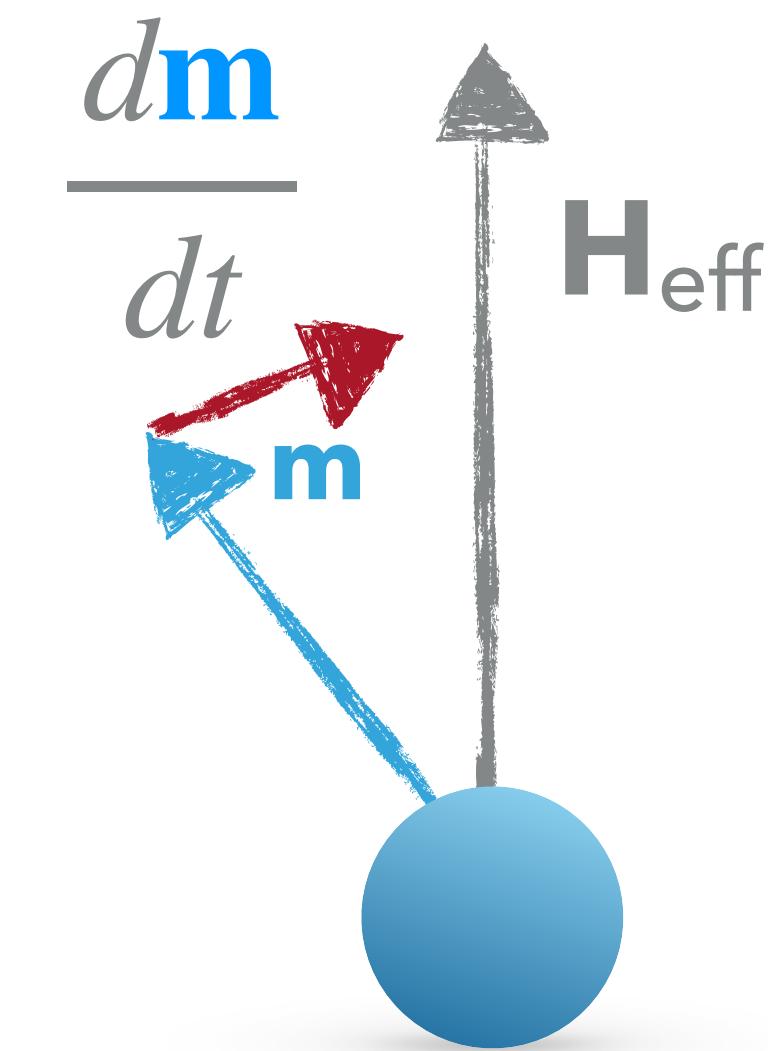
Parameter in
Ubermag is γ_0 ,
not γ .



GILBERT DAMPING

- ▶ Makes \mathbf{m} align with \mathbf{H}_{eff} .
- ▶ Parameter: α

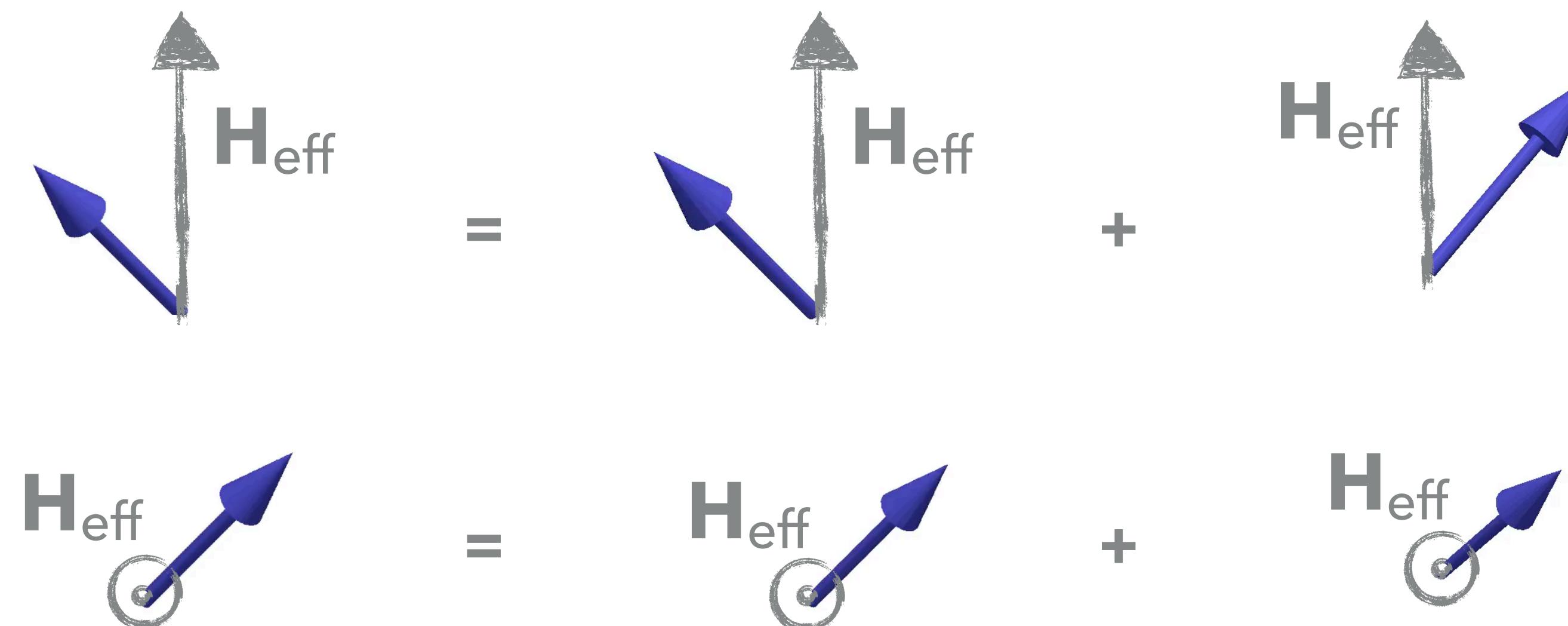
$$\frac{d\mathbf{m}}{dt} = \alpha \left(\mathbf{m} \times \frac{d\mathbf{m}}{dt} \right) \quad (\alpha > 0)$$



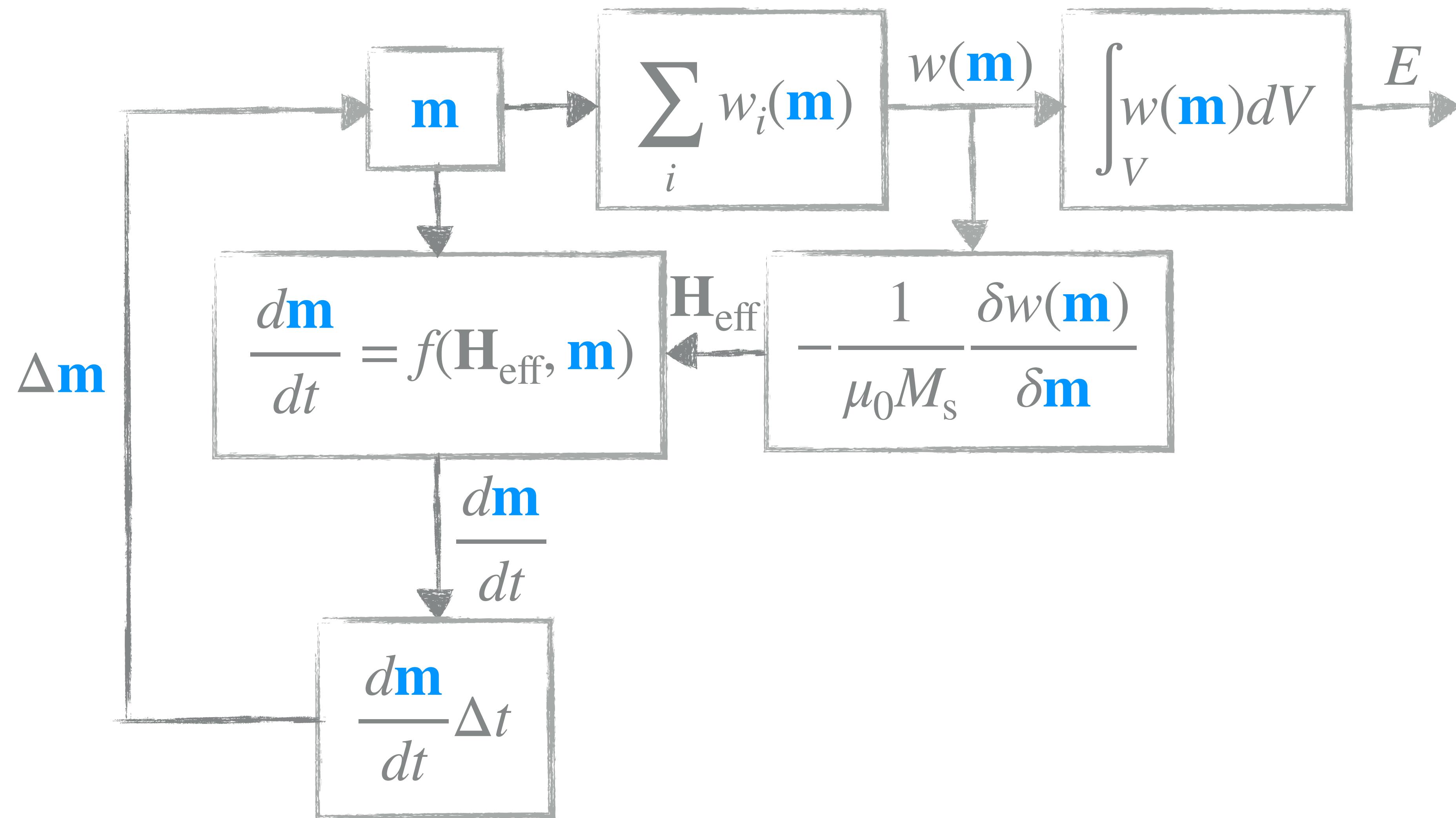
LANDAU-LIFSCHITZ-GILBERT EQUATION

$$\frac{d\mathbf{m}}{dt} = -\gamma_0 \mathbf{m} \times \mathbf{H}_{\text{eff}} + \alpha \left(\mathbf{m} \times \frac{d\mathbf{m}}{dt} \right)$$

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



(OVERSIMPLIFIED) MICROMAGNETIC SIMULATOR



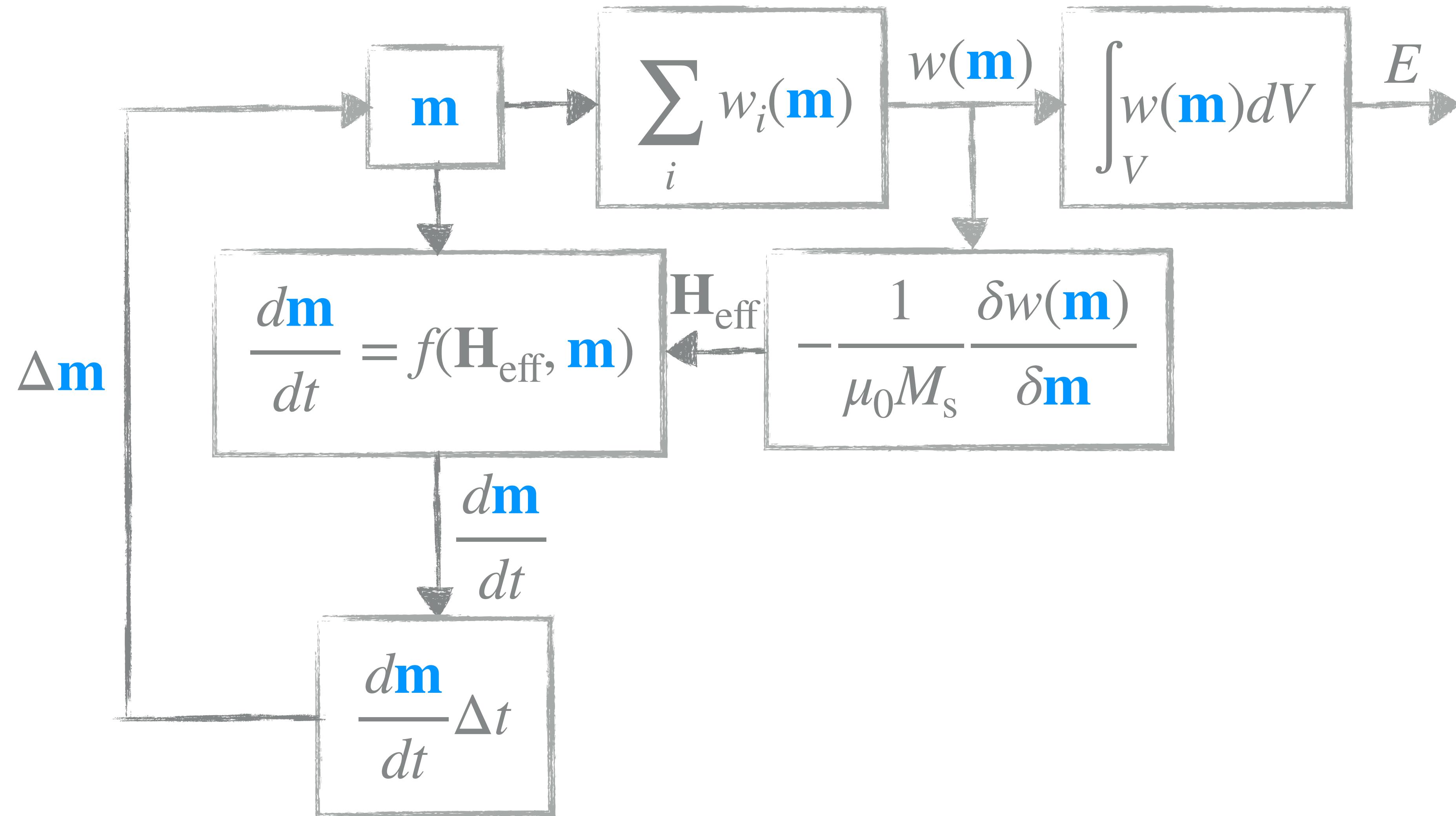


ubermag

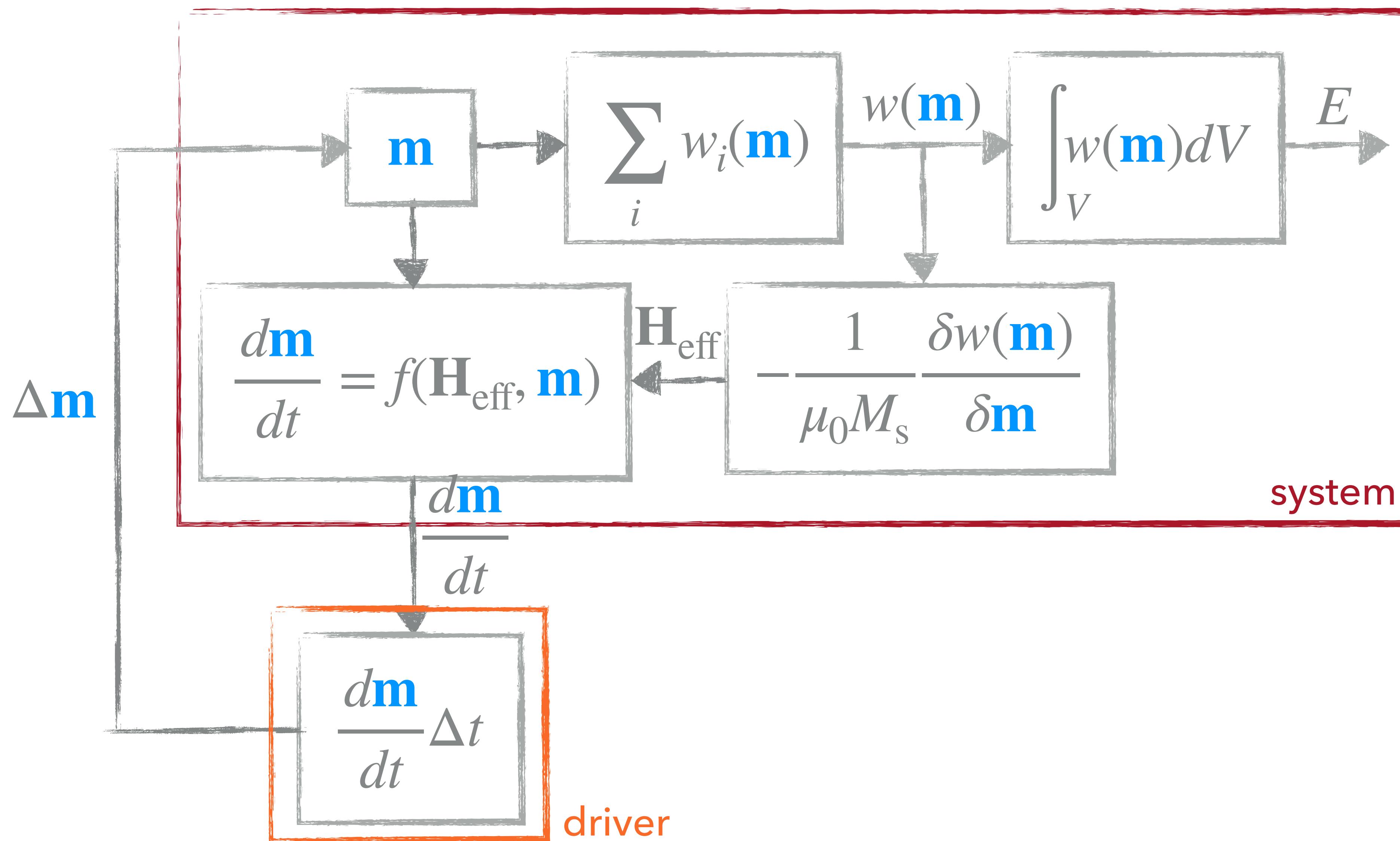
PART 2

**UBERMAG SIMULATION
SKELETON**

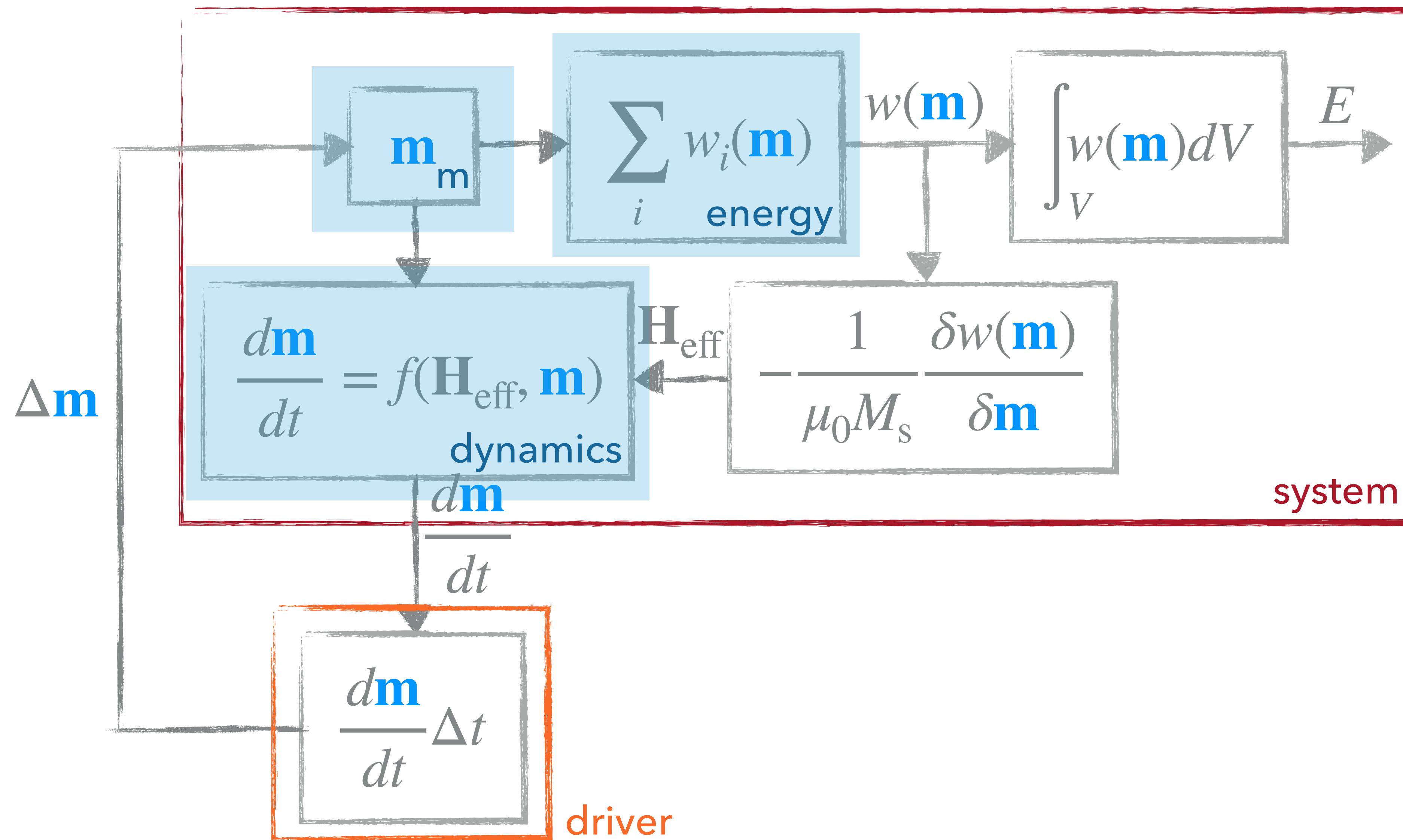
(OVERSIMPLIFIED) MICROMAGNETIC SIMULATOR



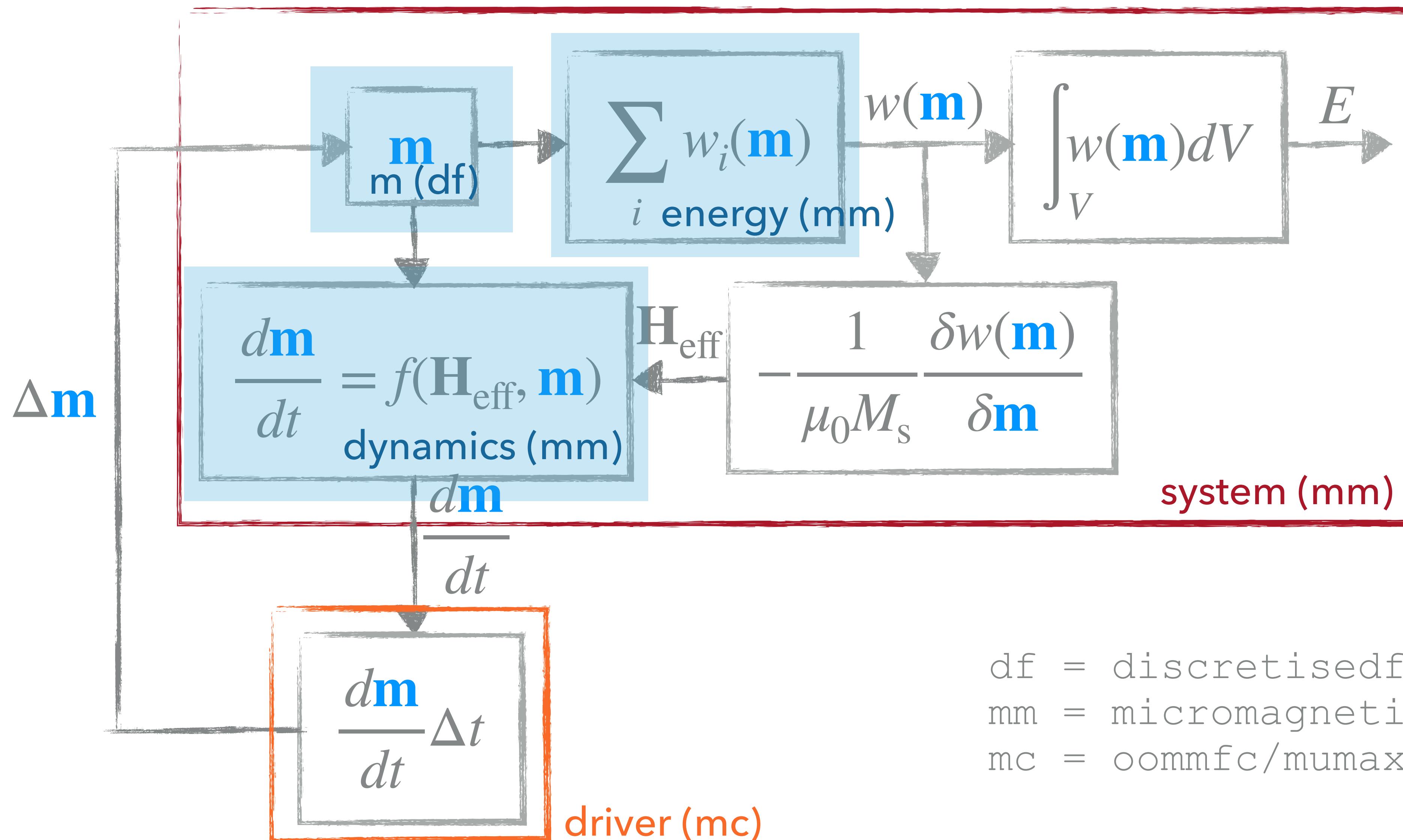
(OVERSIMPLIFIED) MICROMAGNETIC SIMULATOR



(OVERSIMPLIFIED) MICROMAGNETIC SIMULATOR



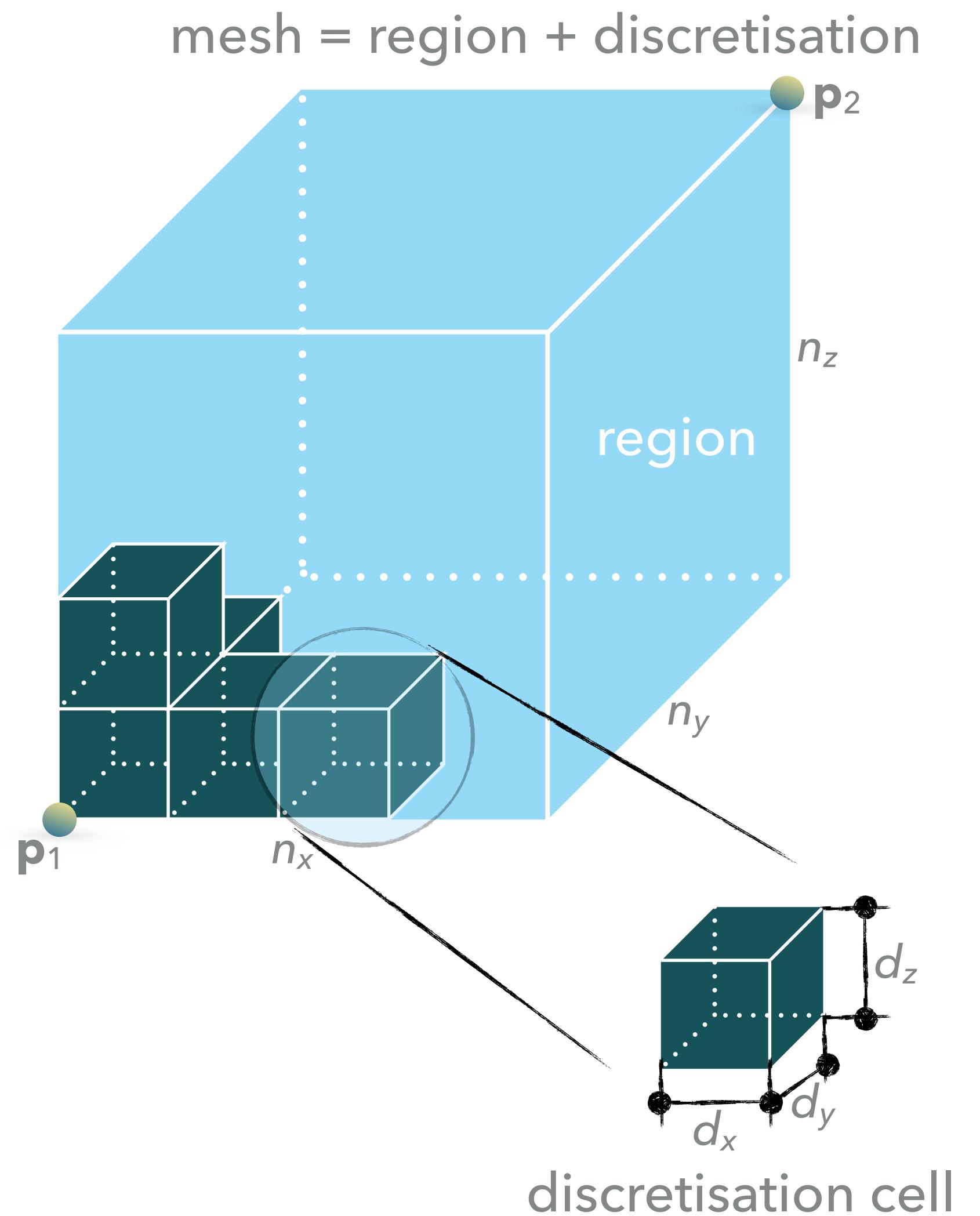
(OVERSIMPLIFIED) MICROMAGNETIC SIMULATOR



df = discretisedfield
 mm = micromagneticmodel
 mc = oommfc/mumax3c

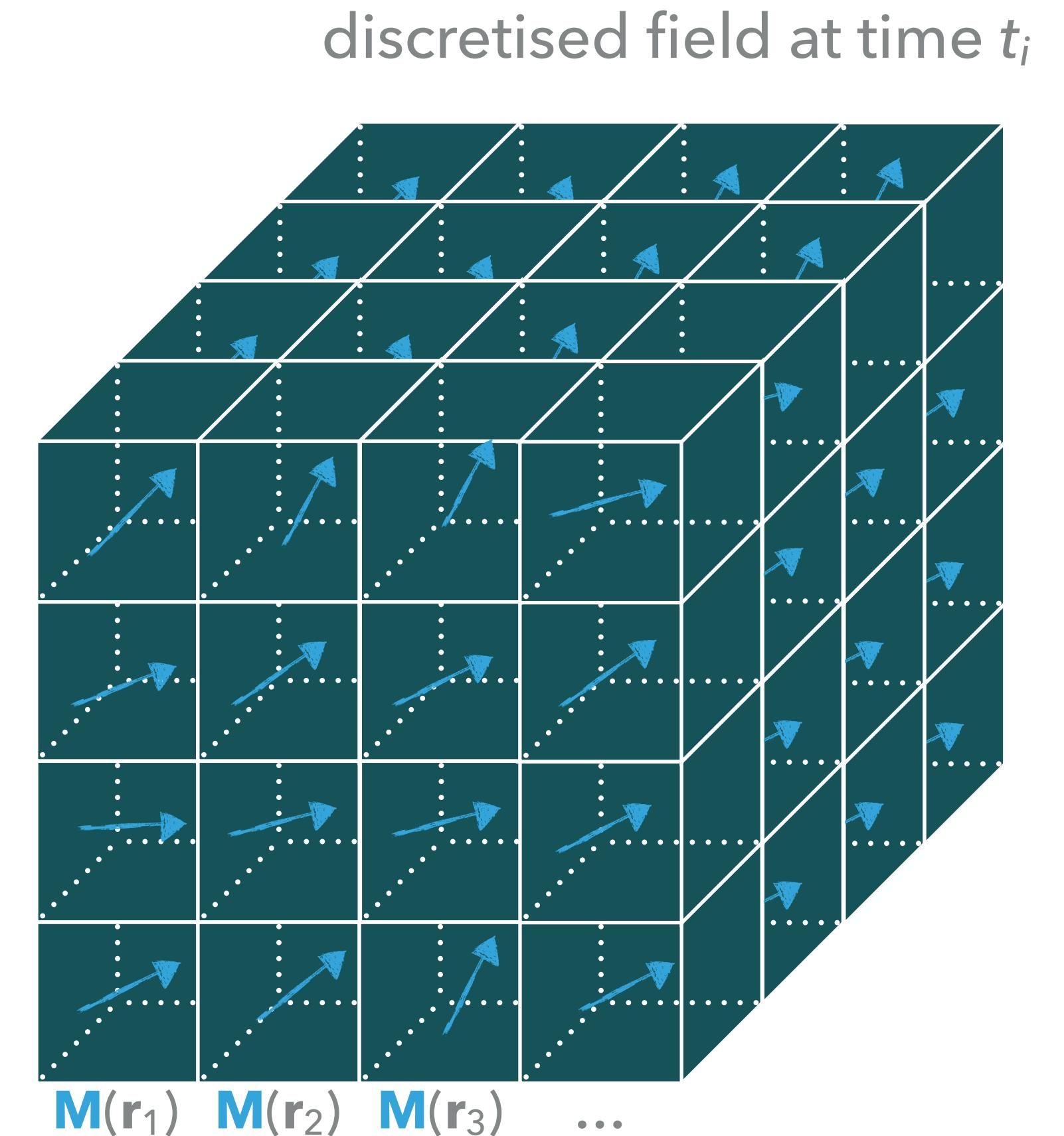
DEFINING MAGNETISATION FIELD 1/2

- ▶ Ubermag package: `discretisedfield`
 - ▶ `import discretisedfield as df`
- ▶ **Region** defines the domain in three-dimensional space.
 - ▶ `region = df.Region(p1=(...), p2=(...))`
- ▶ **Mesh** discretises the region
 - ▶ Either discretisation **cell size or the number of cells** can be defined.
 - ▶ `mesh = df.Mesh(region=region, cell=(...))` or
 - ▶ `mesh = df.Mesh(region=region, n=(...))`



DEFINING MAGNETISATION FIELD 2/2

- ▶ Ubermag package: `discretisedfield`
 - ▶ `import discretisedfield as df`
- ▶ **Field** defines a value at all discretisation cells.
 - ▶ `m = df.Field(mesh=mesh, dim=..., value=..., norm=...)`
- ▶ **Parameters** are:
 1. `mesh (df.Mesh)`
 2. `dim: dimension of the value (int)`
 3. `value: field value (int, iterable, callable)`
 4. `norm: vector field norm (int, callable)`



DEFINING SYSTEM OBJECT

- ▶ Ubermag package: micromagneticmodel
 - ▶ `import micromagneticmodel as mm`
- ▶ First, we define the **system object**:
 - ▶ `system = mm.System(name='my_system_name')`
- ▶ After that, we define:
 - ▶ `system.m = m # the field we defined on previous slide`
 - ▶ `system.energy = # the sum of individual energy terms`
 - ▶ `system.dynamics = ... # the sum of individual dynamics terms`

ENERGY EQUATION

- ▶ Energy equation is a **part of the system object**.
- ▶ The system class and energy term classes are a part of `micromagneticmodel` package.
 - ▶ `import micromagneticmodel as mm`
- ▶ First the **system object** is defined
 - ▶ `system = mm.System(name="system_name")`
- ▶ Then, energy equation is defined
 - ▶ `system.energy = mm.Exchange(A=1e-12) + mm.Demag()`

OVERVIEW OF ENERGY TERMS WE INTRODUCED SO FAR

name	parameters	class	comments
Zeeman	\mathbf{H} (A/m)	mm.Zeeman	parameter is \mathbf{H} not \mathbf{B}
uniaxial anisotropy	K (J/m ³), \mathbf{u}	mm.UniaxialAnisotropy	sign of K , $ \mathbf{u} =1$
exchange	A (J/m)	mm.Exchange	$A < 0$ not justified
DMI (T, O)	D (J/m ²), crystal class	mm.DMI	sign of D
DMI (C_{nv})	D (J/m ²), crystal class	mm.DMI	sign of D
cubic anisotropy	K (J/m ³), $\mathbf{u}_1, \mathbf{u}_2$	mm.CubicAnisotropy	sign of K , $ \mathbf{u}_i =1$
demagnetisation	None	mm.Demag	computationally expensive

DYNAMICS EQUATION

- ▶ Dynamics equation is a part of the system object.
- ▶ The system class and dynamics term classes are a part of `micromagneticmodel` package.
 - ▶ `import micromagneticmodel as mm`
- ▶ First the system object is defined
 - ▶ `system = mm.System(name="system_name")`
- ▶ Then, energy equation is defined
 - ▶ `system.dynamics = Precession(gamma0=.....) + mm.Damping(alpha=...)`

OVERVIEW OF DYNAMICS TERMS WE INTRODUCED SO FAR

name	parameters	class	comments
precession	γ_0	mm.Precession	parameter is γ_0 not γ
damping	α	mm.Damping	$\alpha > 0$
...

DRIVERS

- ▶ There are two main types of drivers and they are a **part of a specific micromagnetic calculator**
 - ▶ `import oommfc as mc`
- ▶ **Minimisation driver**
 - ▶ iterates until $\mathbf{m} \times \mathbf{H}_{\text{eff}} = 0$
 - ▶ `md = oc.MinDriver()`
- ▶ **Time driver**
 - ▶ iterates until the maximum time is reached
 - ▶ `td = oc.TimeDriver()`
- ▶ After the driver is defined, it is applied to the system object using **drive method**
 - ▶ `td.drive(system)`

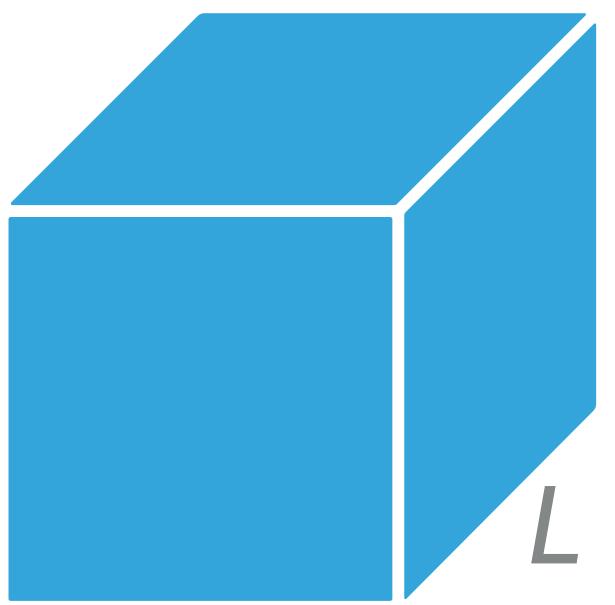


PART 3

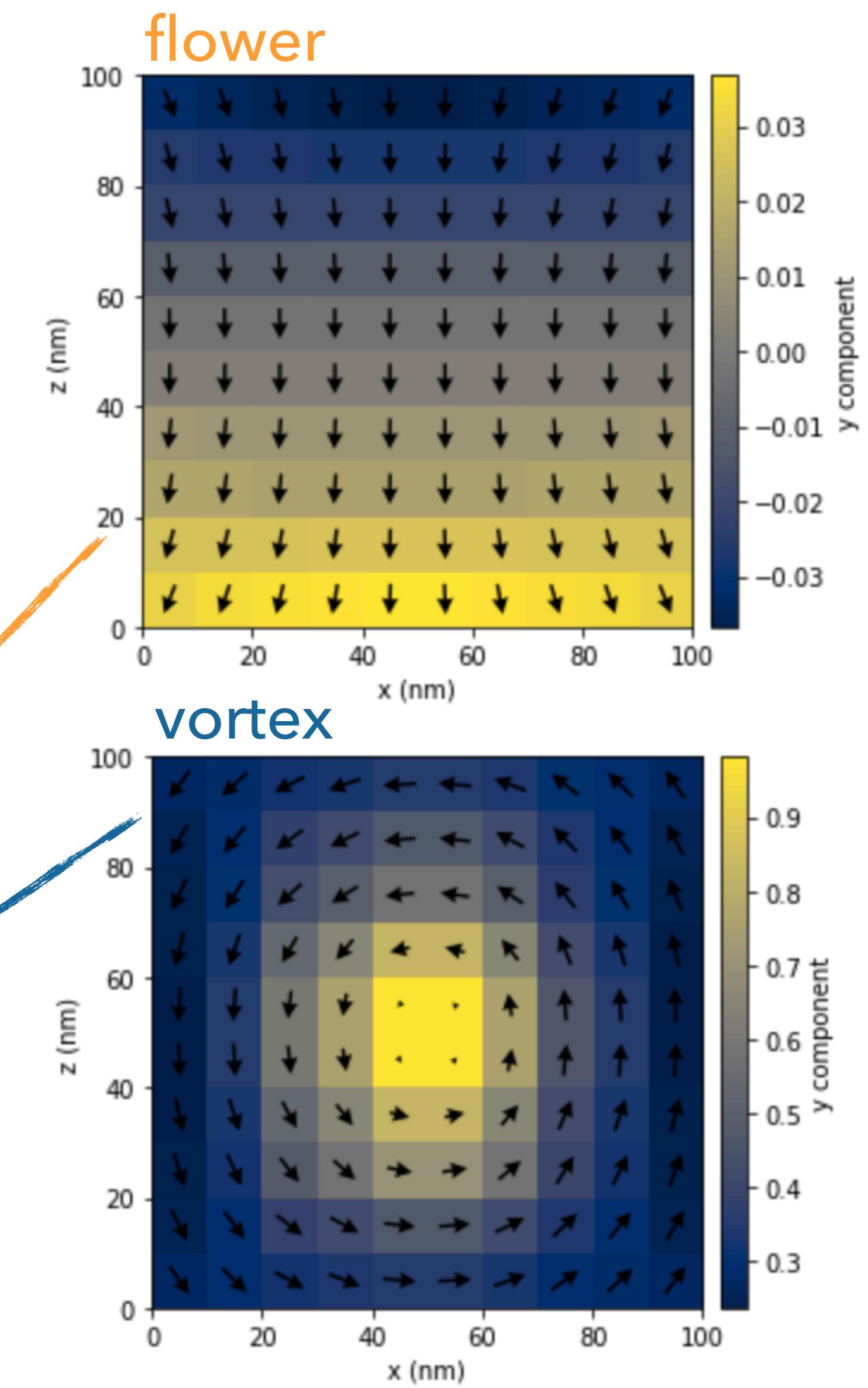
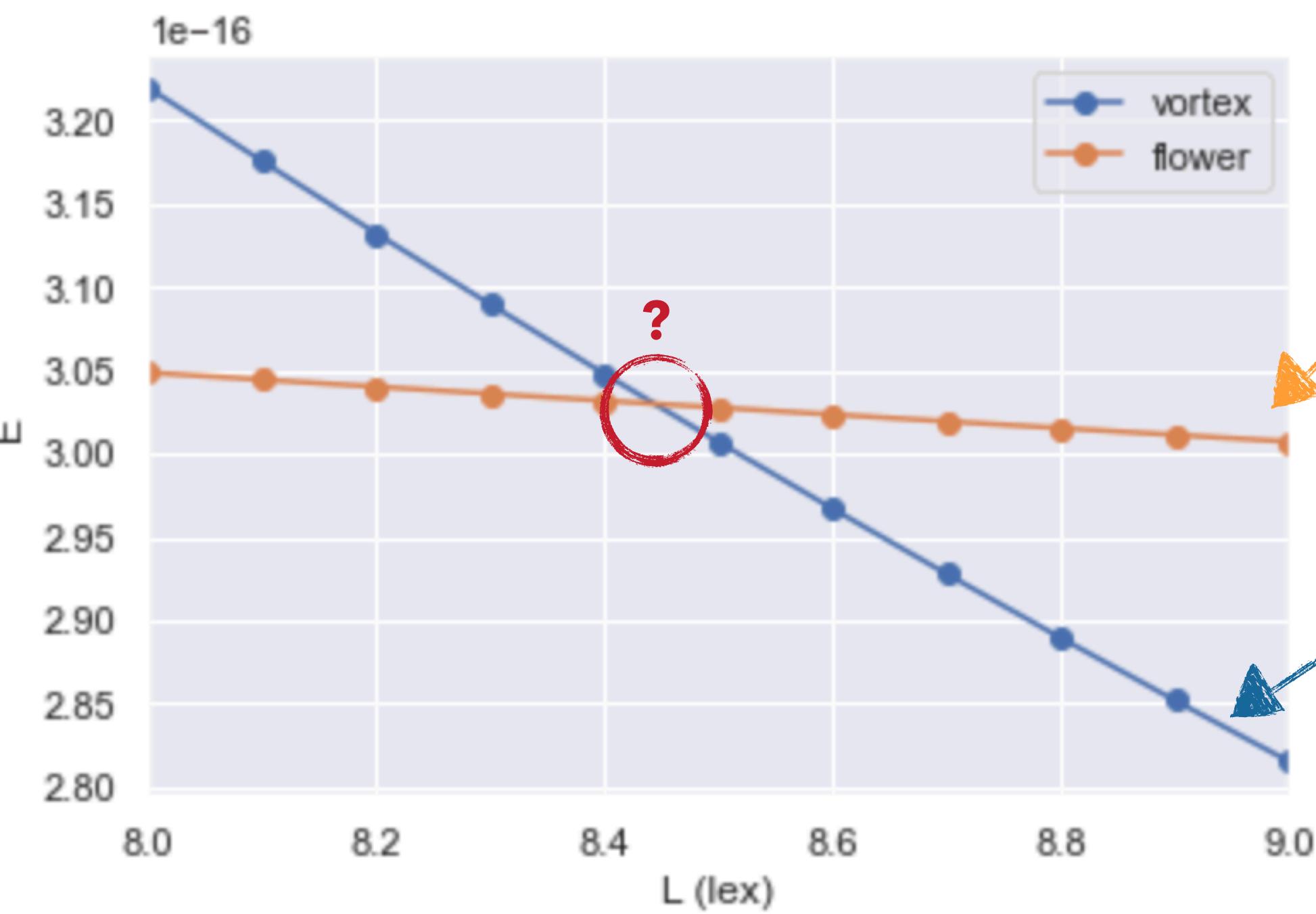
**COMPUTATIONAL
WORKFLOW**

PART 3 - COMPUTATIONAL WORKFLOW

CASE STUDY: STANDARD PROBLEM 3



Research question: For what cube edge length L , vortex and flower states have the same energy?



PART 3 - COMPUTATIONAL WORKFLOW

STEP 1: WRITE CONFIGURATION FILE

```
my_project — IPython: Users/mb4e10 — emacs -nw stdprob3.mif — 95x37
# 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

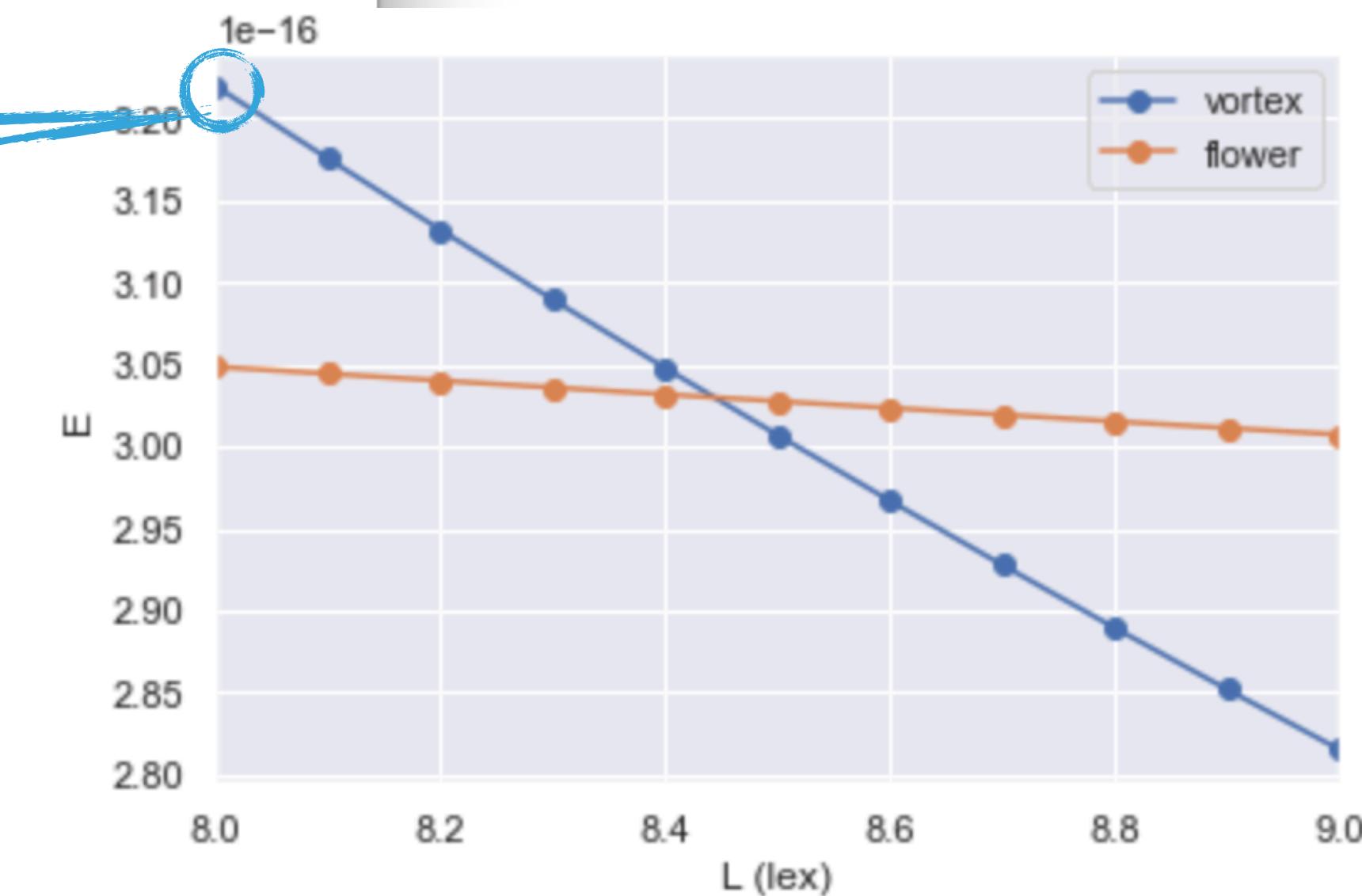
Parameter initial_state "vortex", Initial state should be
## one of "uniform", "vortex", "cant", "cantvortex", "twisted",
## "random" or "file <filename>"; in the last case <filename> is the
## name of a file to use as the initial configuration.

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 and
# exchange length based on parameters L and N.
-uu---F1 stdprob3.mif Top L1 (Fundamental)
```



PART 3 - COMPUTATIONAL WORKFLOW

STEP 2: RUN SIMULATION

1. configuration file

2. run OOMMF

3. output file

The screenshot shows a terminal window titled "my_project — IPython: Users/mb4e10 — bash → python — 95x37". The window displays the following command and its execution:

```
Marijans-MBP:my_project mb4e10$ ls  
stdprob3.mif  
Marijans-MBP:my_project mb4e10$ tclsh $00MMFTCL boxsi +fg stdprob3.mif -exitondone 1  
Start: "/Users/mb4e10/my_project/stdprob3.mif"  
Options: -exitondone 1 -threads 2  
Boxsi version 1.2.1.0  
Running on: marijans-macbook-pro.local  
OS/machine: Darwin/x86_64  
User: mb4e10 PID: 72176  
Number of threads: 2  
Mesh geometry: 32 x 32 x 32 = 32 768 cells  
Checkpoint file: /Users/mb4e10/my_project/sp3-vortex-seed0000.restart  
Boxsi run end.  
Marijans-MBP:my_project mb4e10$ ls  
sp3-vortex-seed0000.out stdprob3.mif  
Marijans-MBP:my_project mb4e10$
```

Annotations with blue circles highlight specific parts of the terminal output:

- A circle around "stdprob3.mif" under step 1.
- A circle around the command "tclsh \$00MMFTCL boxsi +fg stdprob3.mif -exitondone 1" under step 2.
- A circle around "sp3-vortex-seed0000.out" under step 3.

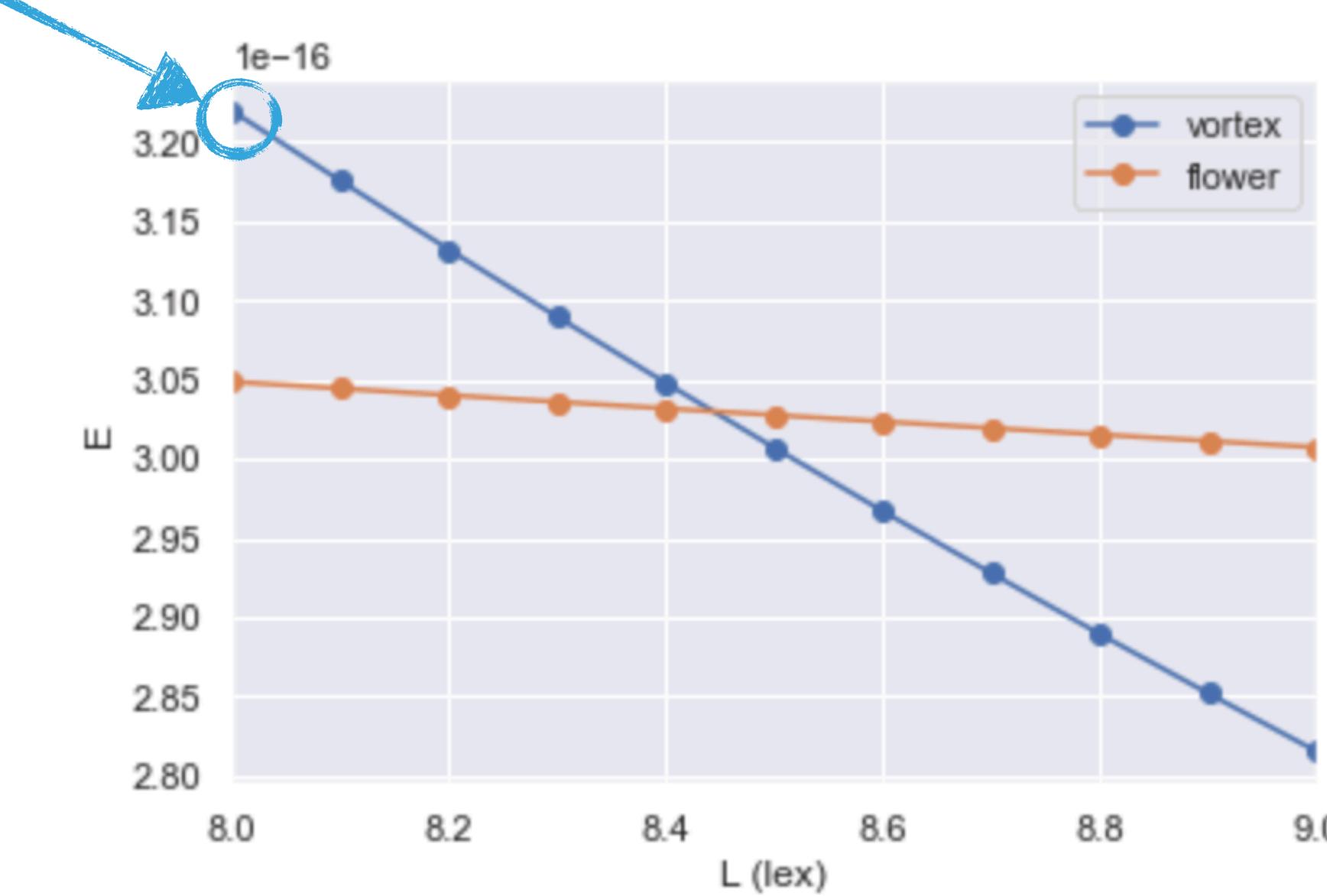
PART 3 - COMPUTATIONAL WORKFLOW

STEP 3: READ RESULTS

PART 3 - COMPUTATIONAL WORKFLOW

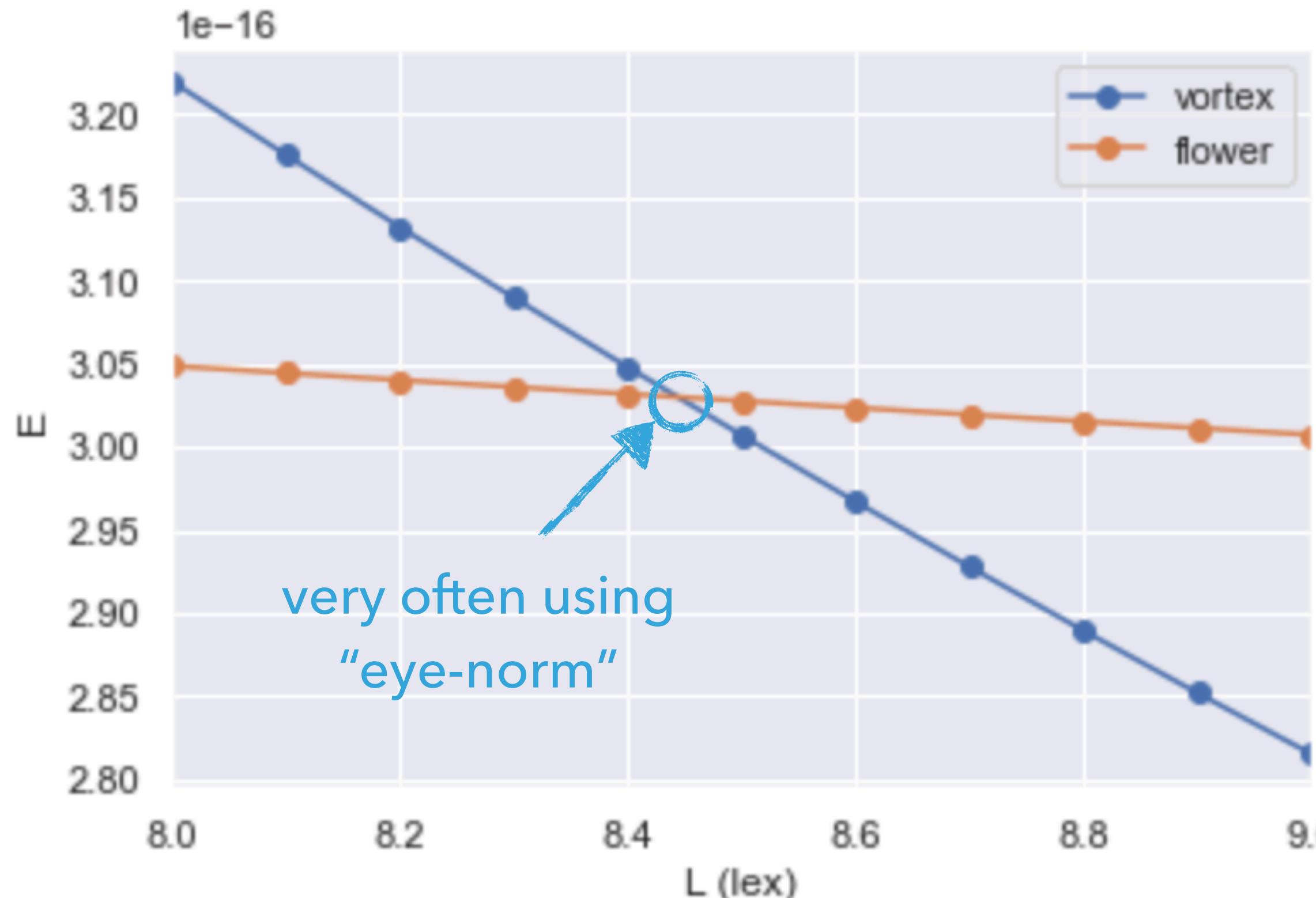
LOOP THROUGH STEPS 1, 2, 3...

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	?	?

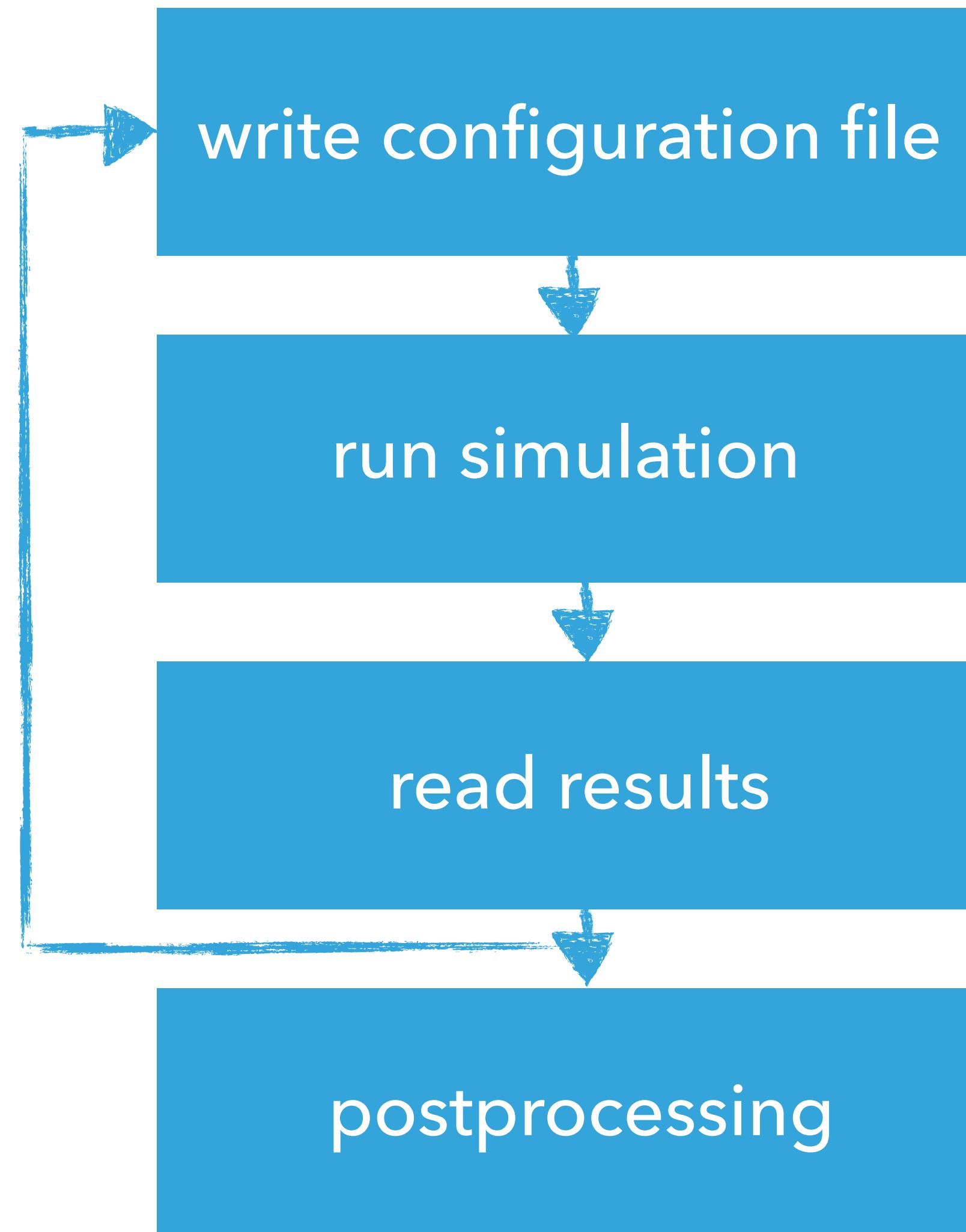


LAST STEP: POSTPROCESSING

- ▶ After we obtained all data points, we **plot the results and find crossing**.
- ▶ For this step, we often use **separate** plotting scripts or graphical user interface (GUI).

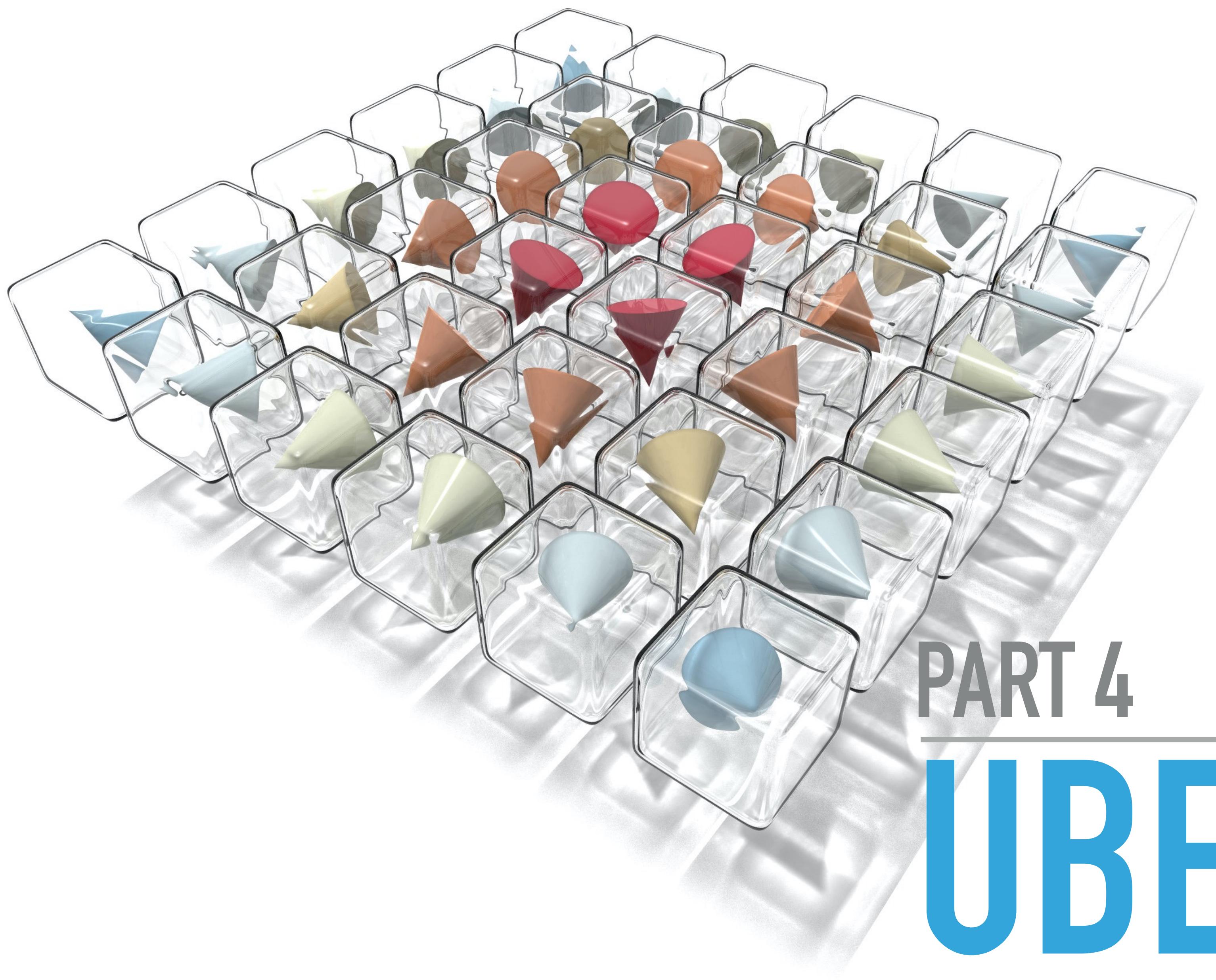


WORKFLOW SUMMARY



WHAT COULD BE THE PROBLEMS WITH THIS WORKFLOW?

1. **Time** consuming
 - ▶ It requires a lot of user input
2. **Keeping log** of all steps that were run and in what order
 - ▶ I clicked here, then I changed that, then I fixed that...
3. Separate **postprocessing scripts**
 - ▶ Every group has their own scripts with different dependencies
4. **Sharing** the exact workflow
5. **Reproducibility**
6. **Very difficult to automate**
7. Very **steep learning curve**

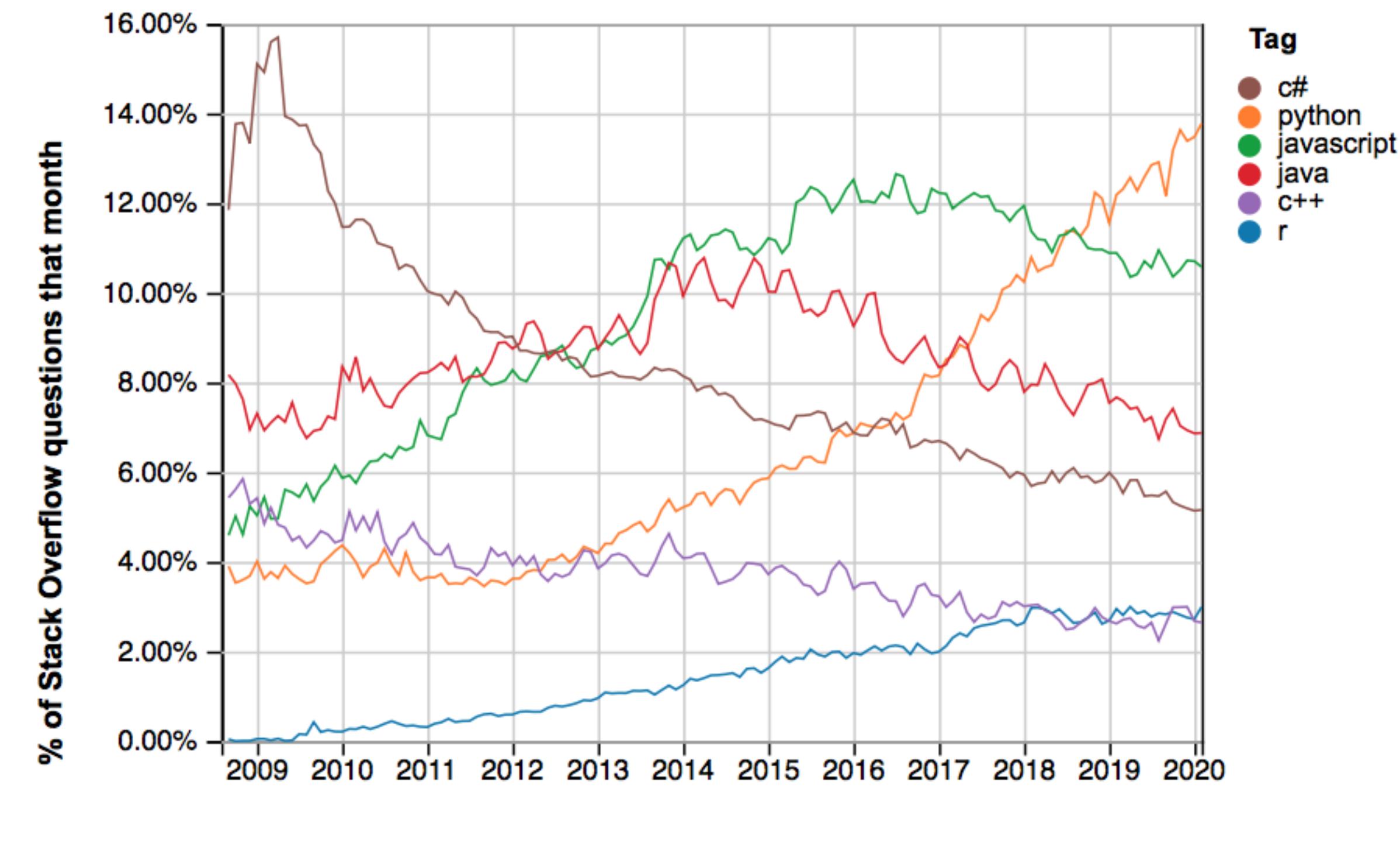


PART 4

UBERMAG

WHY DID WE CHOOSE PYTHON?

- ▶ **Modern** programming language
- ▶ The language core is **easy to read and easy to learn**
 - ▶ Easily accessible to students
 - ▶ Increasingly popular in **software engineering**
 - ▶ The most popular in **computational and data science**
 - ▶ Very **well documented** and well supported
 - ▶ Interpreted language
 - ▶ www.python.org



Source: <https://towardsdatascience.com>

SCIENTIFIC PYTHON ECOSYSTEM

- ▶ **numpy**: linear algebra
- ▶ **scipy**: numerical analysis
- ▶ **matplotlib**: 2d (and some 3d) plotting
- ▶ **pandas**: big data for Python
- ▶ **scikit-learn**: machine learning
- ▶ **Jupyter Notebook**
- ▶ No need to reinvent the wheel.

JUPYTER

- ▶ **Executable document**
- ▶ Text, equations, images, code, and results in a **single document**
- ▶ Easily **shared**
- ▶ Easily **reproducible**
- ▶ Hosted in **web browser**
- ▶ Can be **run in the cloud** (Binder)
- ▶ www.jupyter.org

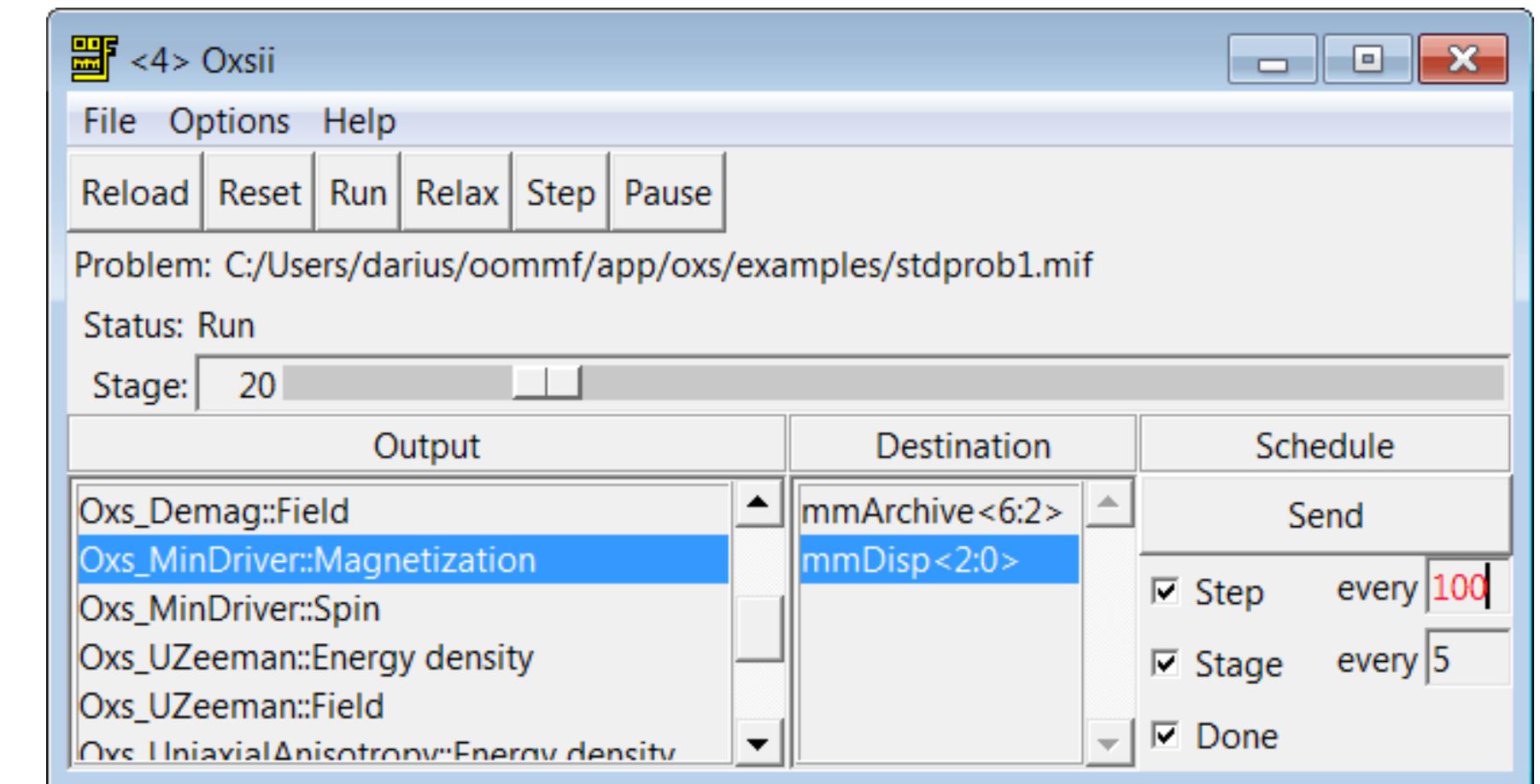


UBERMAG

*"... provides Python interface to OOMMF
and mumax3 (for now), exposes
micromagnetic simulations to Python's
scientific ecosystem, and embeds them into
Jupyter notebook."*

WHY DID WE START WITH OOMMF?

- ▶ Probably **the most widely used** micromagnetic simulation tool
- ▶ Developed at **NIST**, US, **since ~1998** by Michael Donahue & Don Porter
- ▶ **Cited over 2200** times in scientific publications
- ▶ Written in **C++ & Tcl**
- ▶ **Finite-difference** code
- ▶ Very often **used for comparisons** between codes
- ▶ <https://math.nist.gov/oommf/>



```
stdprob3.mif
26 #####
27 # Auxiliary variables:
28
29 # Work out Ms so magnetostatic energy density, Km=0.5*mu0*Ms^2,
30 # is 1e6 J/m^3
31 set Km 1e6
32 set Ms [expr {sqrt(2*$Km/$mu0)}]
33
34 # Arbitrarily set cube dimension to 100 nm, and compute cellsize and
35 # exchange length based on parameters L and N.
36 set cubesize 100e-9 ;# Cube dimension in meters
37 set cellsize [expr {$cubesize/$N}] ;# In meters
38 set lex [expr {$cubesize/$L}] ;# exchange length
39
40 # Set K1 to 0.1*Km
41 set K1 [expr {$Km/10.}]
42
43 # Compute A so that cubesize is requested number of exchange lengths
44 set A [expr {0.5*$mu0+$Ms*$Ms*$lex*$lex}] ;# Exchange coefficient, J/m
45
46 #####
47 Report "A=$A, K1=$K1, Ms=$Ms, lex=$lex, L=$L, seed=$seed"
48
49 #####
50 # Tcl script for CantedVortex proc
51 #
52 # Coordinate transform to select initial vortex orientation:
53 proc CantedVortexInit { vec } {
54     proc Mag { v } {
55         set v0 [lindex $v 0]
56         set v1 [lindex $v 1]
57         set v2 [lindex $v 2]
```

DOCUMENTATION

- ▶ **GitHub** organisation: <https://github.com/ubermag>
- ▶ Each package has an **online documentation**:
 - ▶ [package_name].[readthedocs.io](#)
- ▶ There is **API Reference** which is always up-to-date and covers 100% of functionality.
- ▶ **Tutorials as Jupyter notebooks** provide the summary of functionalities in a certain context
 - ▶ **Available in the cloud** - Binder
 - ▶ Does not cover all functionality

SUPPORT

- ▶ Functionality requests, bug reports, general support
- ▶ The preferred way is to raise an issue in **help repository**:
 - ▶ <https://github.com/ubermag/help>
 - ▶ There is a **permanent record** of communication
 - ▶ There is a chance, somebody already asked a similar question
 - ▶ **Do not ask confidential questions!**
 - ▶ Please **be patient** (it's only me behind all the codes)

SUPPORT THE SOFTWARE

1. M. J. Donahue and D. G. Porter. OOMMF User's Guide, Version 1.0, Interagency Report NISTIR 6376. National Institute of Standards and Technology, Gaithersburg, MD (Sept 1999).
2. 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>

LICENCE

- ▶ BSD-3-Clause
 - ▶ You can do with the code whatever you want
- ▶ Obligations
 - ▶ You have to cite
 - ▶ If you modify the code, you must copy the licence file and acknowledge the source

CONTRIBUTIONS ARE WELCOME

- ▶ **We welcome all types of contributions**
 - ▶ New extensions
 - ▶ Documentation
 - ▶ Tutorials
 - ▶ YouTube videos
 - ▶ Bug fixes
 - ▶ ...
- ▶ If there is anything you would like to contribute to, but you are not sure how, please let us know.