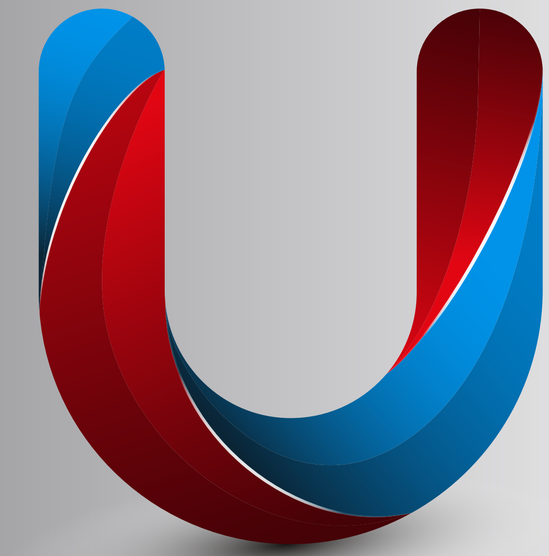
WARNING: This is a presentation PDF and does not have animations. For a MacOS Keynote presentation, ask Marijan.



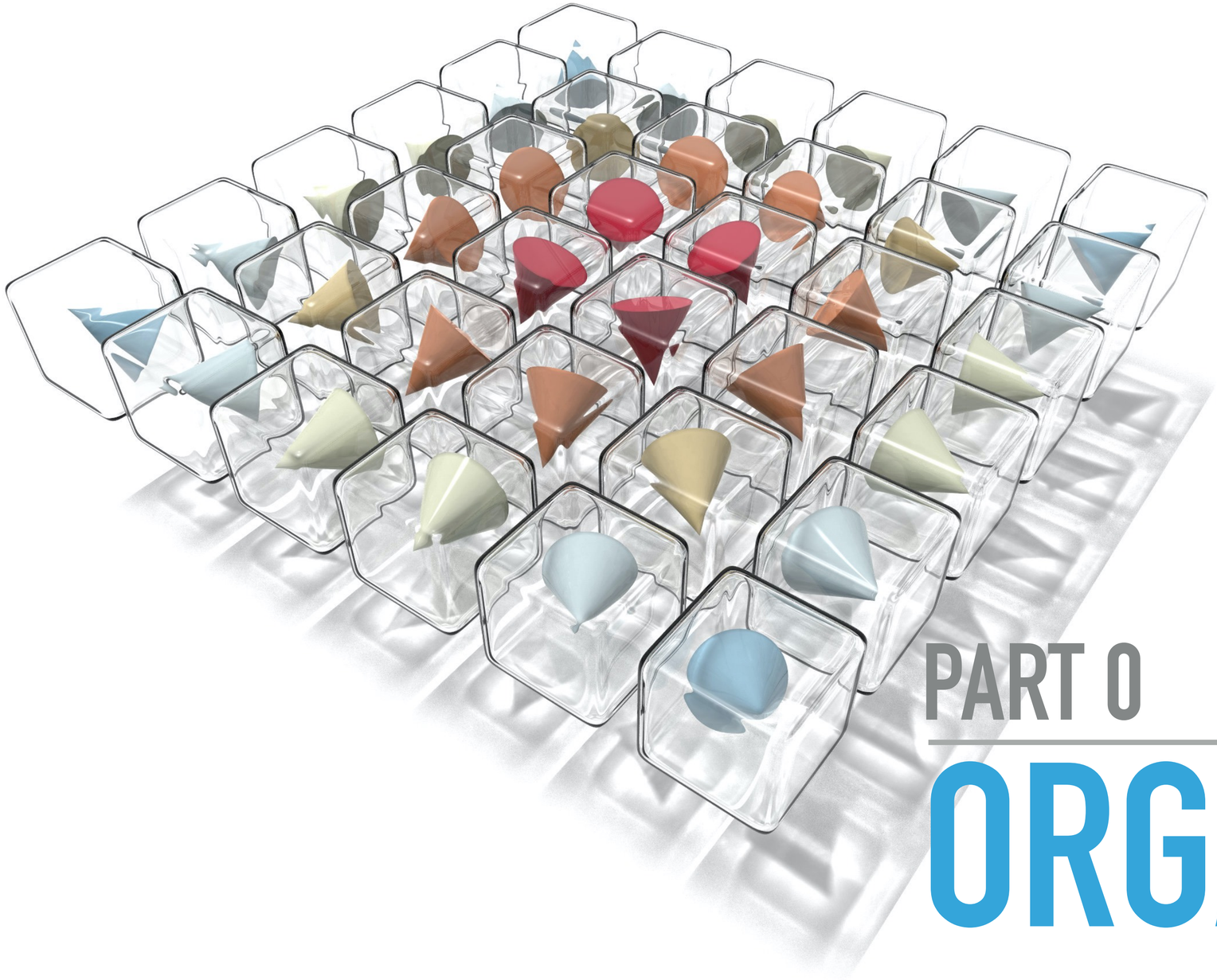
ubermag

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

MICROMAGNETICS WITH UBERMAG

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

Last updated: 18 June 2020



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. Basics of micromagnetics, Jupyter, and Python
        - **Thursday 18 June 2020, 12:00-13:30** (ET)
    2. 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)

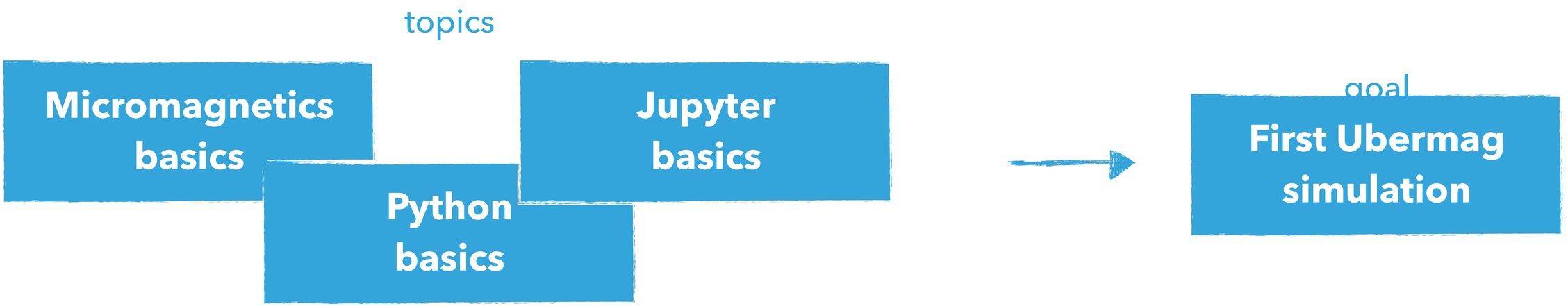
- Ubermag is a **Python package** and it can be installed in many different ways (pip, miniconda, anaconda-navigator, docker…)
- We recommend the installation using **miniconda**:

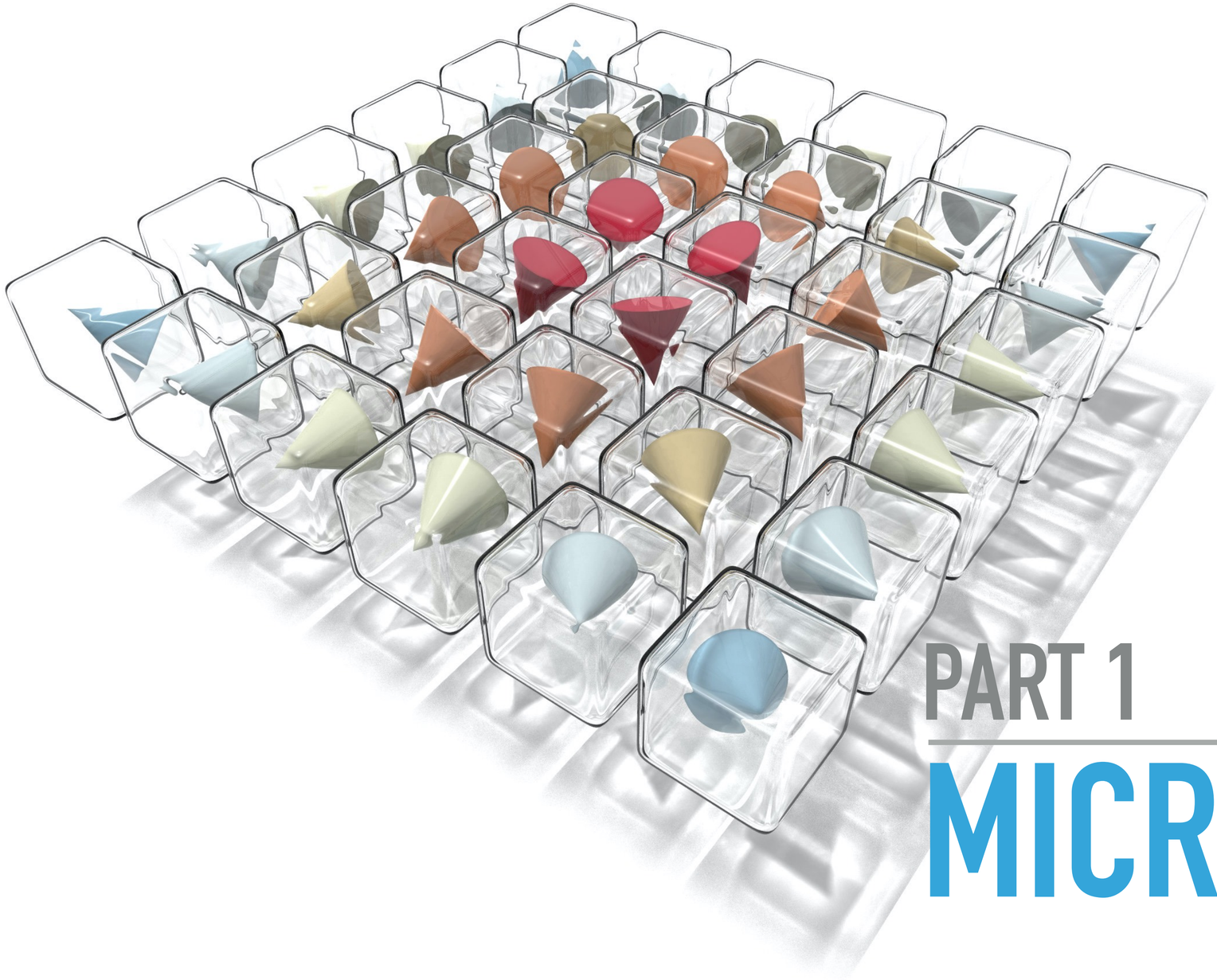
    1. Install Miniconda
    2. `conda install —channel conda-forge ubermag`

- 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).





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 **M**(**r**, *t*) is a function of both space **r** and time *t*.

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

MICROMAGNETIC ASSUMPTIONS

1. Magnetisation field **M**(**r**, *t*) is **differentiable** (continuous and slowly changing) with respect to both space **r** and time *t*.
2. The magnetisation field **norm is constant** (time invariant).
   - Constant norm |**M**| is represented by **saturation magnetisation** M_s.

$$M_s = |\mathbf{M}| = \text{const.}$$

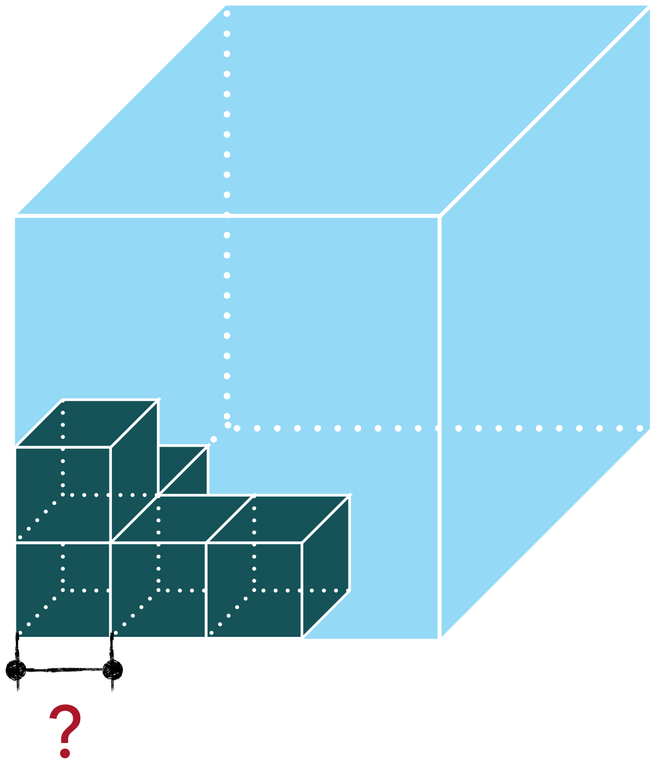
   - Very often, magnetisation is represented by a normalised magnetisation field **m**(**r**, t).

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

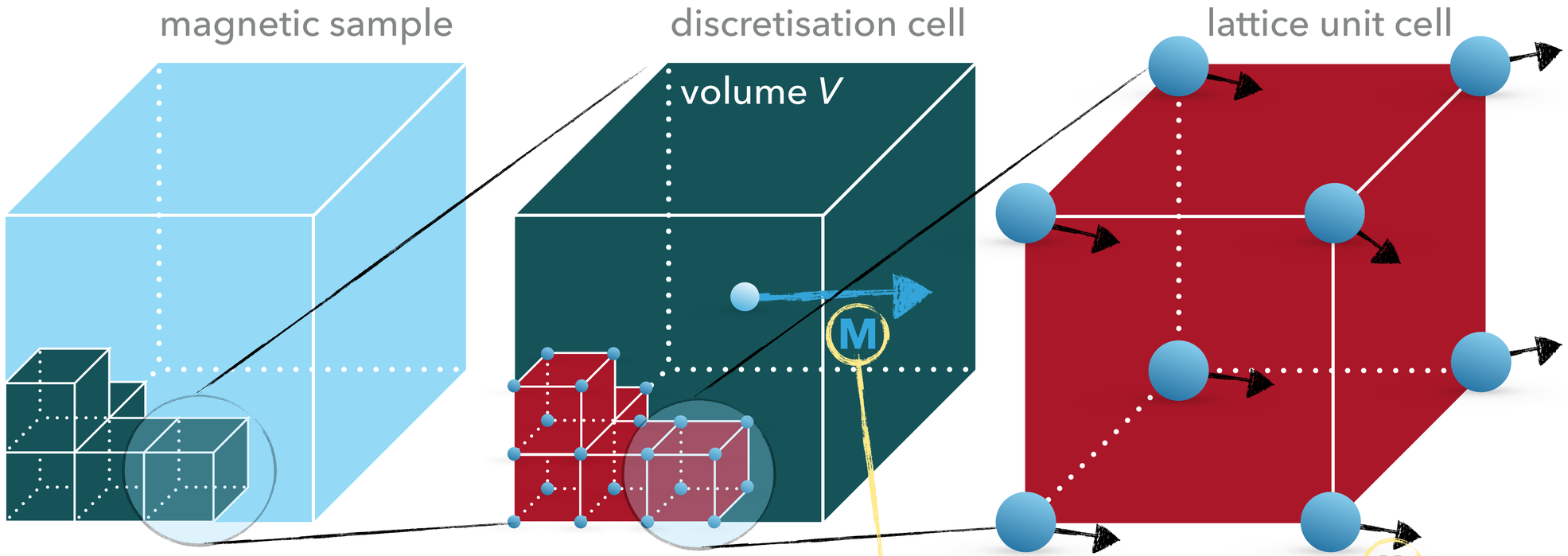
$$|\mathbf{m}| = m_x^2 + m_y^2 + m_z^2 = 1$$

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



$\mathbf{M}(\mathbf{r})$

magnetisation field

discretisation cell coordinate

$$\mathbf{M} = \frac{\sum_{i \in V} \boldsymbol{\mu}_i}{V}$$

DISCRETISED MAGNETISATION FIELD

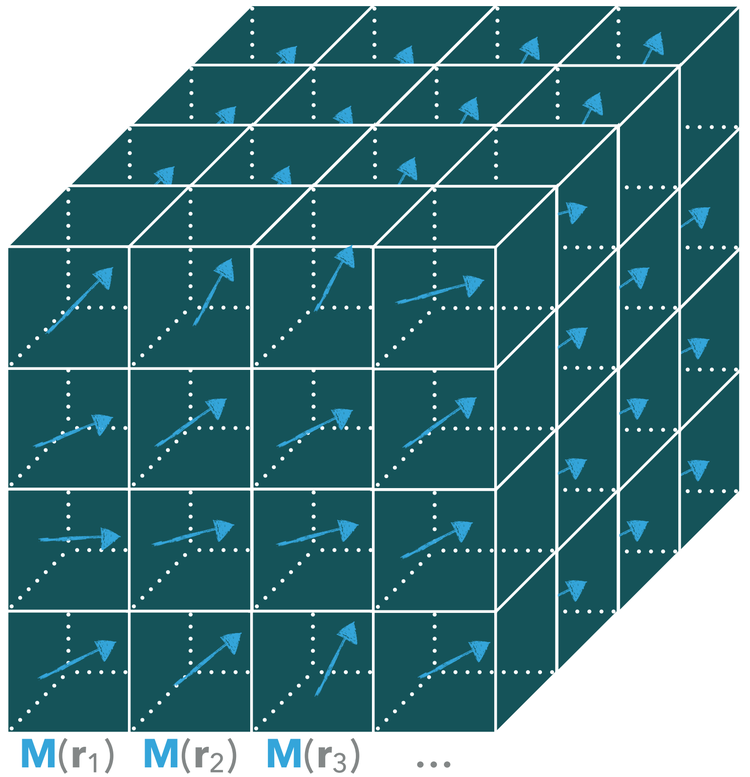
- In continuum approximation, magnetisation is considered to be a **continuous vector field.**
- Magnetisation **M**(**r**, t) is a function of both space **r** and time t.

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

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

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

discretised field at time t_i



ENERGY EQUATION (HAMILTONIAN)

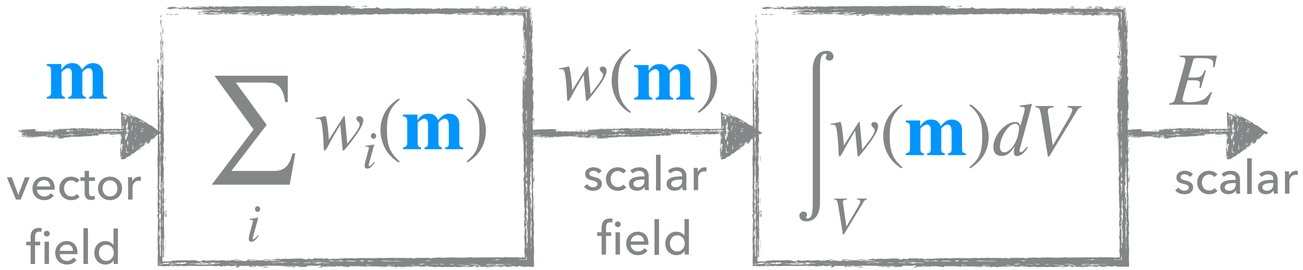
- … is a mapping of magnetisation field **m**=**m**(**r**, *t*) to energy density (scalar) field.

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

user-defined

- By integrating *w*(**m**) over the entire sample volume *V*, the energy functional is

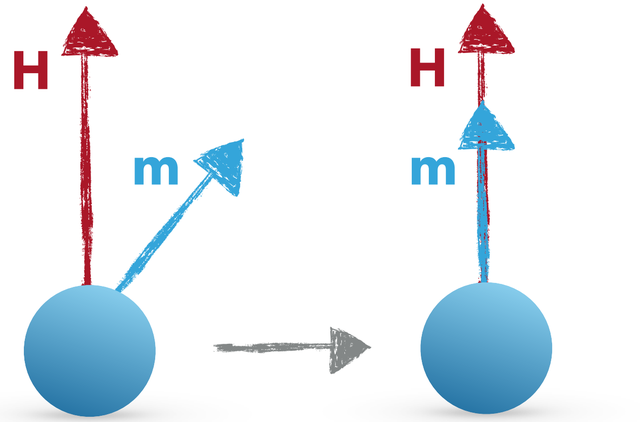
$$E[\mathbf{m}] = \int_V w(\mathbf{m}) dV$$



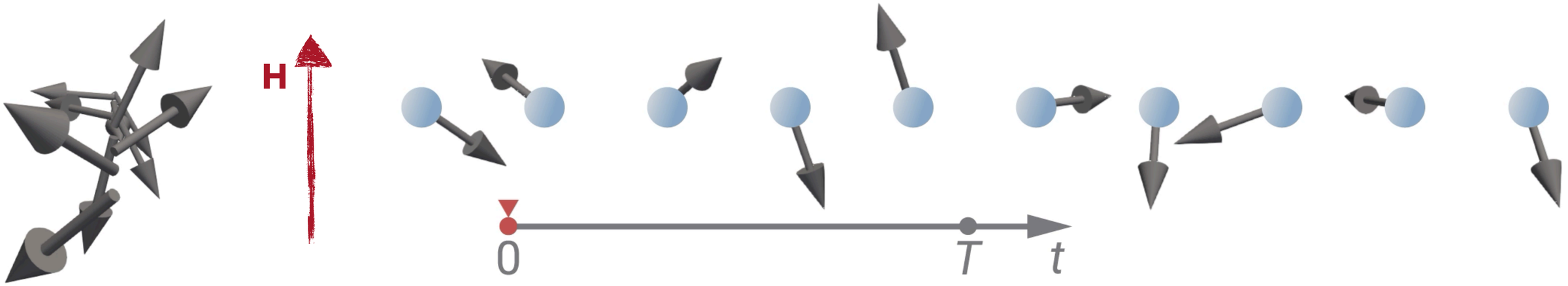
ZEEMAN

- Aligns **m** parallel to **H**.
- Parameter: **H** (A/m)

Parameter in ubermag is **H** and not **B**.



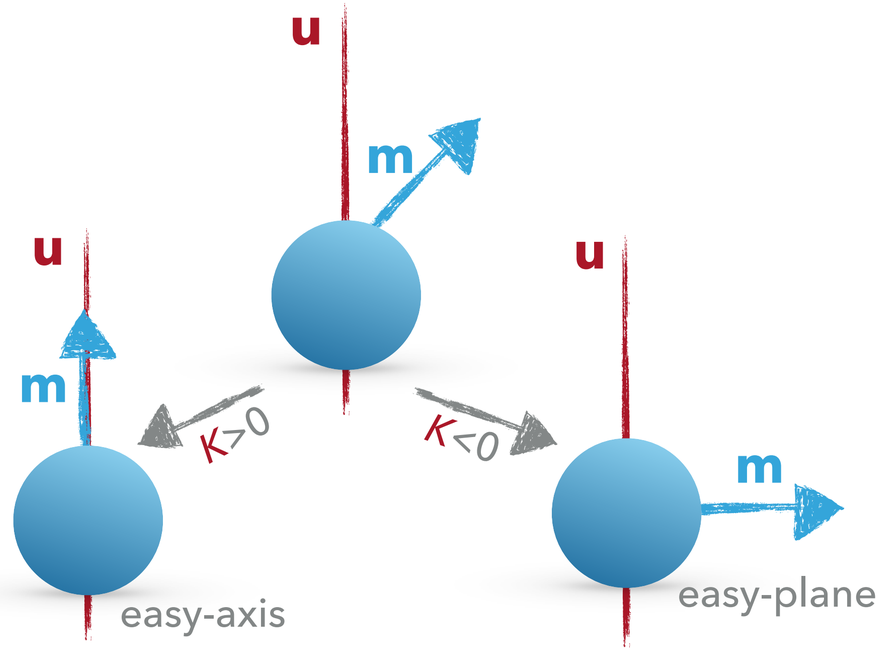
$$w_z = -\mathbf{M} \cdot \mathbf{B} \qquad (\mathbf{B} = \mu_0 \mathbf{H},\ \mathbf{M} = M_s \mathbf{m})$$
$$= -\mu_0 M_s \mathbf{m} \cdot \mathbf{H}$$



UNIAXIAL ANISOTROPY

- Aligns **m** along **u** or perpendicular to **u**.
- Parameters: K (J/m^3), **u**

$$w_{\mathrm{ua}} = -K(\mathbf{m}\cdot\mathbf{u})^2 \qquad (|\mathbf{m}|=1,\ |\mathbf{u}|=1)$$
$$= -K\cos^2\theta$$
$$= -K(1-\sin^2\theta) = \underbrace{-K}_{\text{const.}} + K\sin^2\theta$$



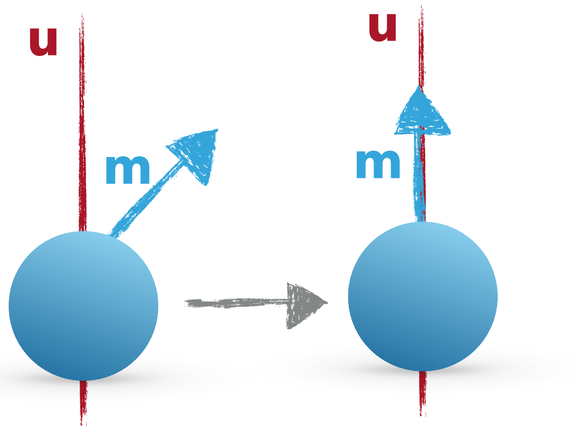
- There are several mathematically equivalent ways of writing uniaxial anisotropy.
- Changing on the sign of K, changes the “character” of uniaxial anisotropy.

Be careful about the sign of K and if |**u**|=1.

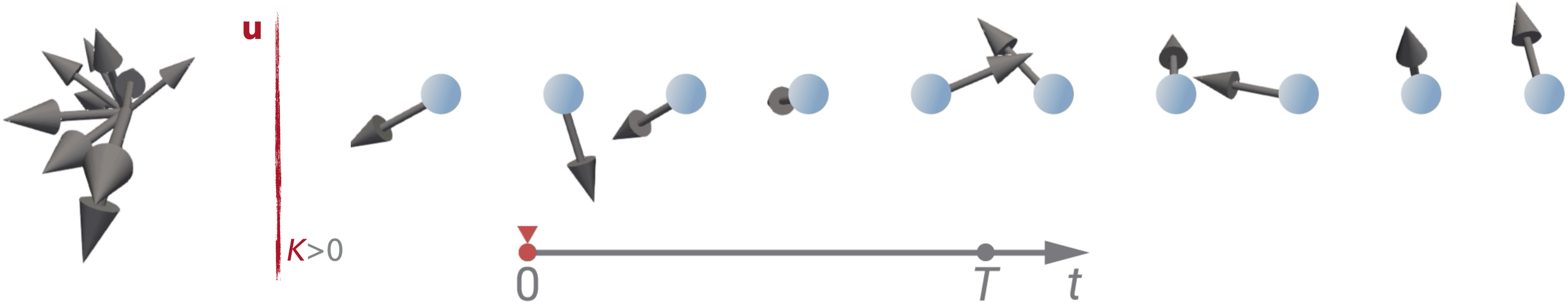
EASY AXIS UNIAXIAL ANISOTROPY

- Aligns **m** parallel to antiparallel to **u**.
- Parameters: K>0 (J/m³), **u**

Be careful that K>0 and |**u**|=1.



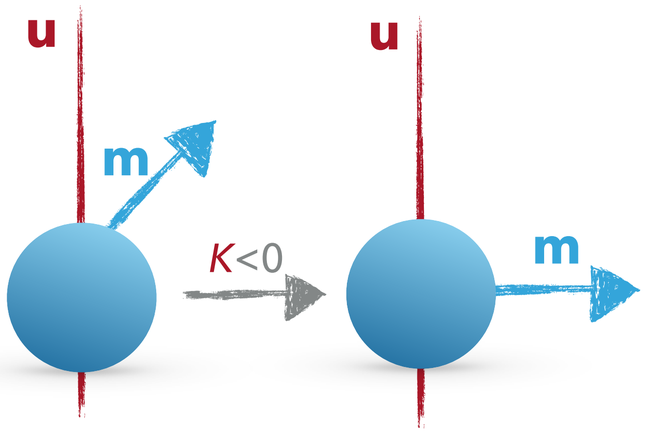
$$w_{\mathrm{ua}} = -K(\mathbf{m}\cdot\mathbf{u})^2 \qquad (|\mathbf{m}| = 1,\ |\mathbf{u}| = 1)$$



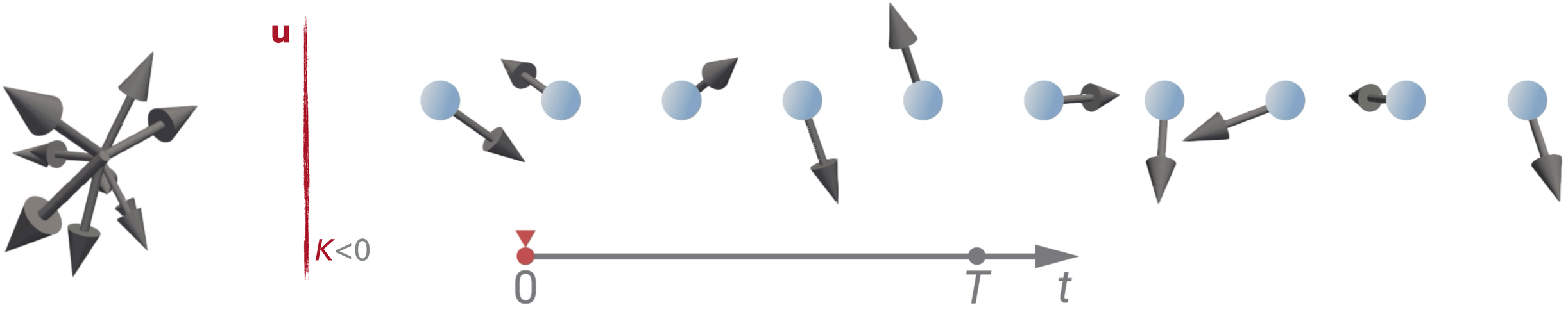
EASY PLANE UNIAXIAL ANISOTROPY

- Aligns **m** perpendicular to **u**.
- Parameters: K<0 (J/m³), **u**

Be careful that K<0 and |**u**|=1.



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



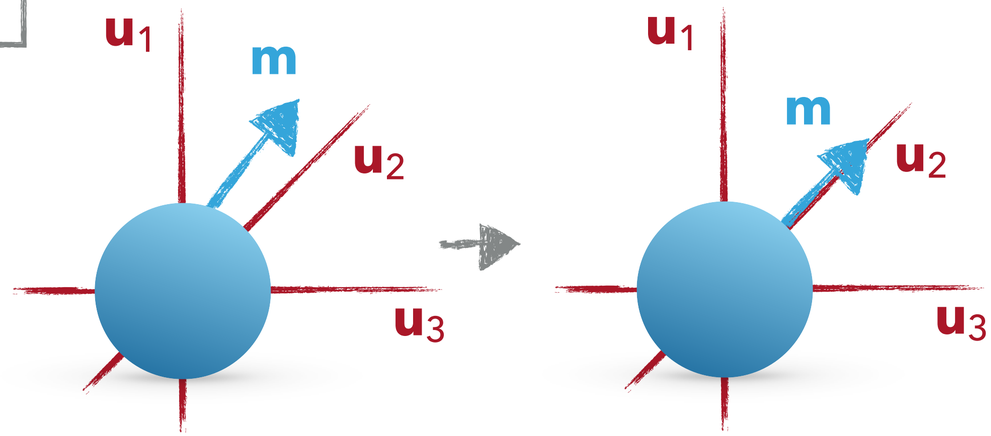
CUBIC ANISOTROPY

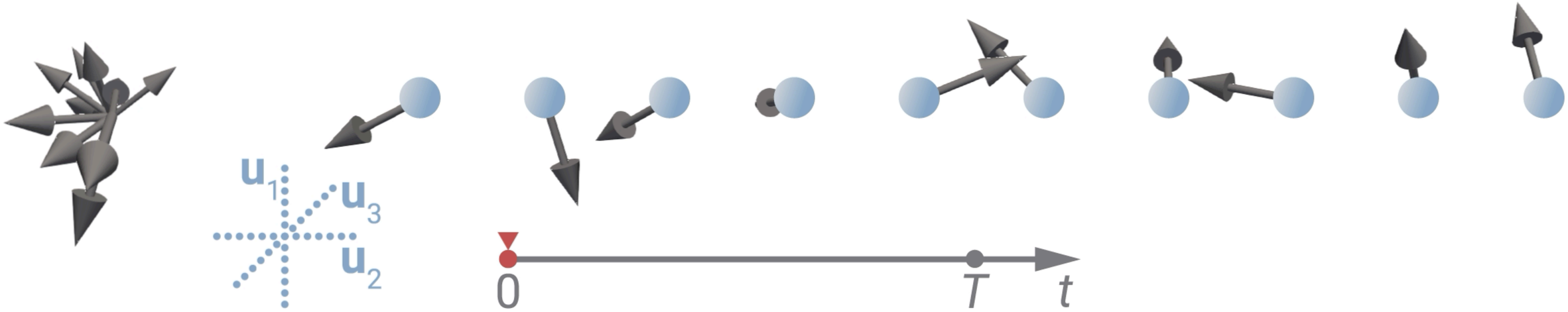
- Aligns **m** along $\mathbf{u}_1$, $\mathbf{u}_2$, or $\mathbf{u}_3$.
- Parameters: K (J/m³), $\mathbf{u}_1$, $\mathbf{u}_2$, (and) $\mathbf{u}_3$

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

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

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





(FERROMAGNETIC) EXCHANGE

- Aligns all magnetic moments (in **m**) parallel to each other.
- Parameter: A (J/m)

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

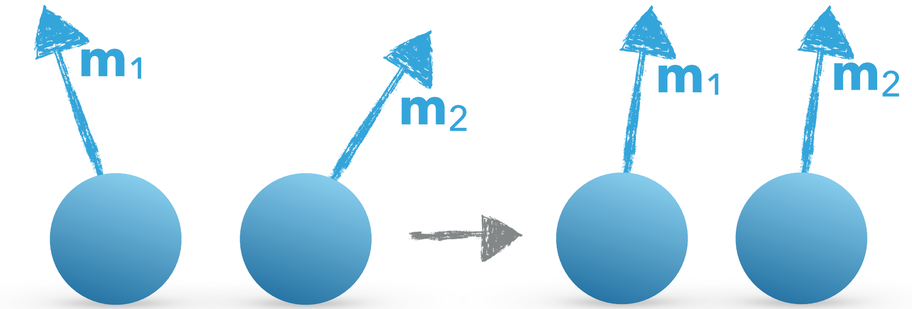
vector Laplacian

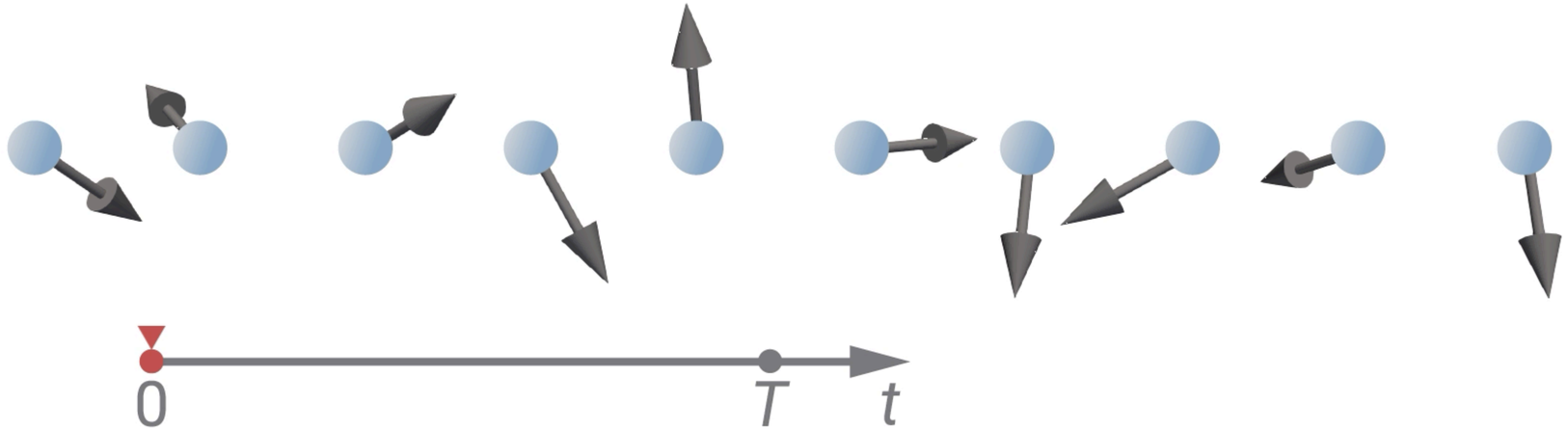
$$w_{\text{ex}} = A\,\mathbf{m}\cdot\nabla^2\mathbf{m} \quad \left(\nabla^2\mathbf{m} = \nabla^2 m_x\hat{\mathbf{x}} + \nabla^2 m_y\hat{\mathbf{y}} + \nabla^2 m_z\hat{\mathbf{z}}\right)$$

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

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

just a convention



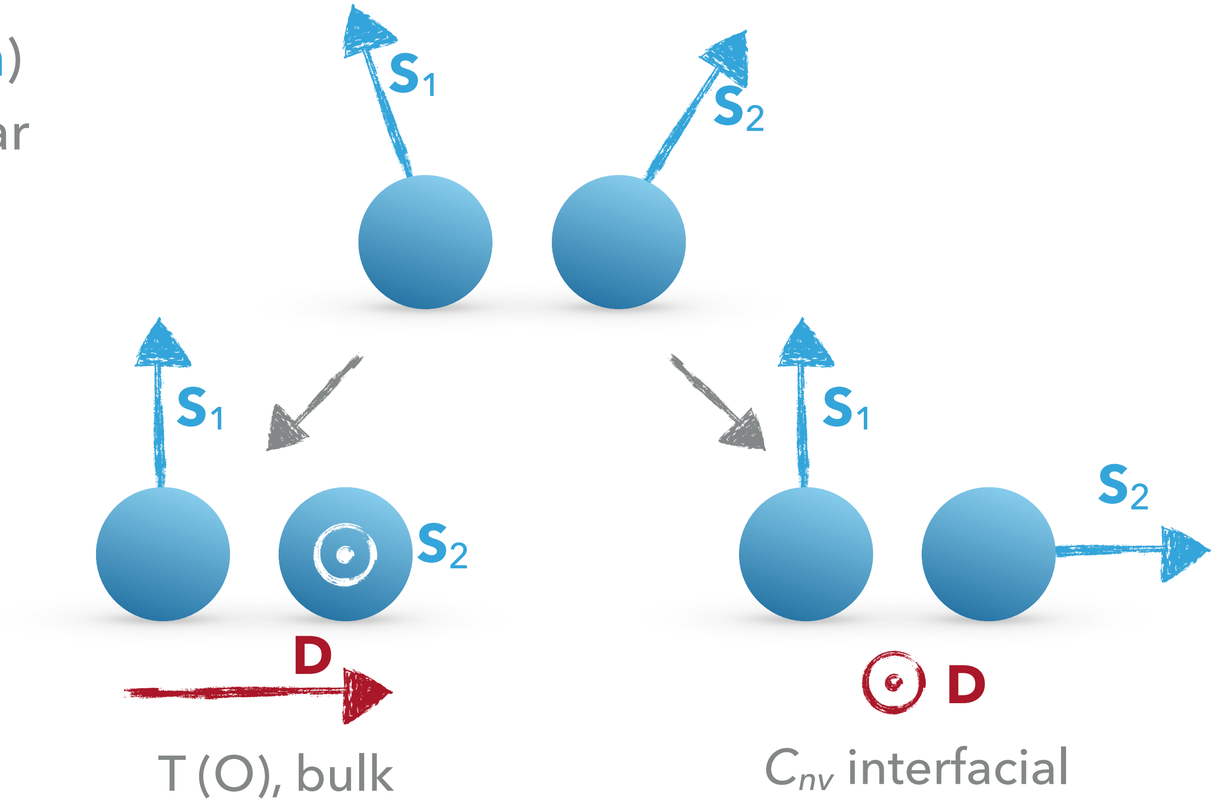


(ATOMISTIC) DZYALOSHINSKII-MORIYA

- Aligns neighbouring magnetic moments (in **m**) perpendicular to each other and perpendicular to **D**.

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

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



T (O), bulk

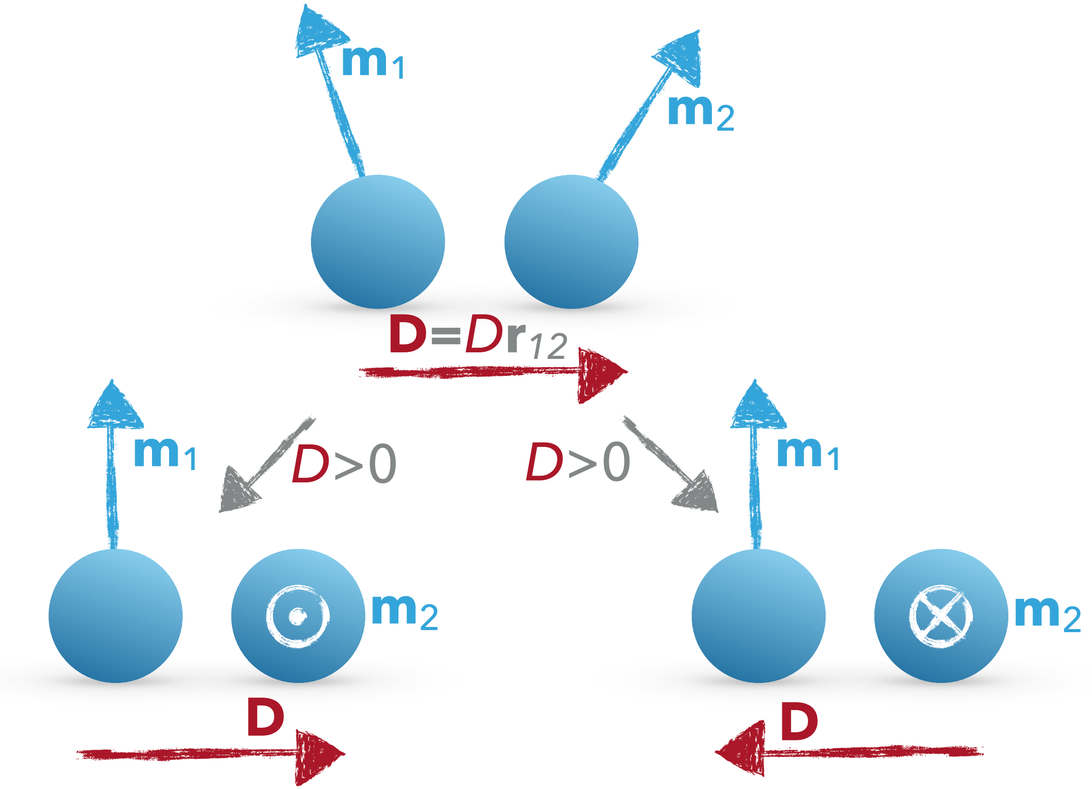
C_{nv} interfacial

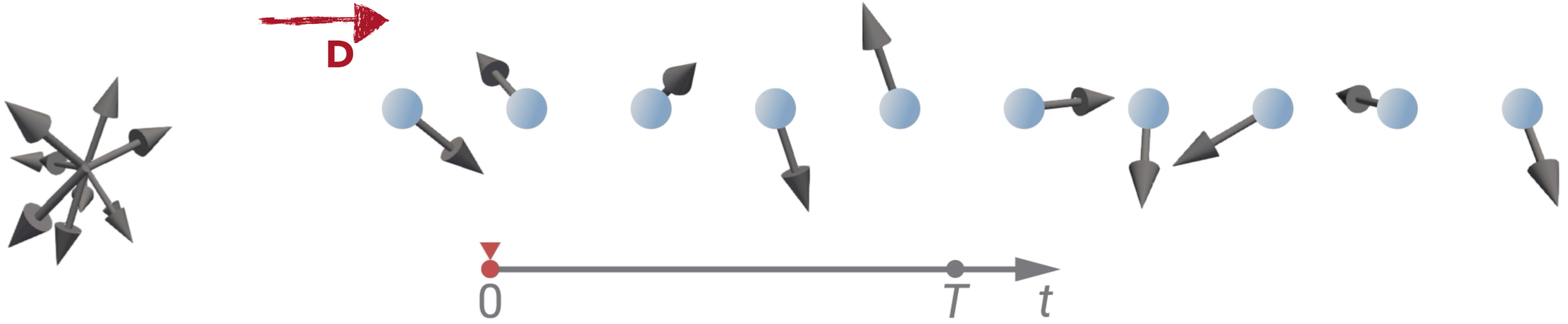
DZYALOSHINSKII–MORIYA (T, 0)

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

$$w_{\mathrm{dmi}} = (\pm) D\mathbf{m} \cdot (\nabla \times \mathbf{m}) \qquad \left(\mathbf{D} = D\mathbf{r}_{ij}\right)$$

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





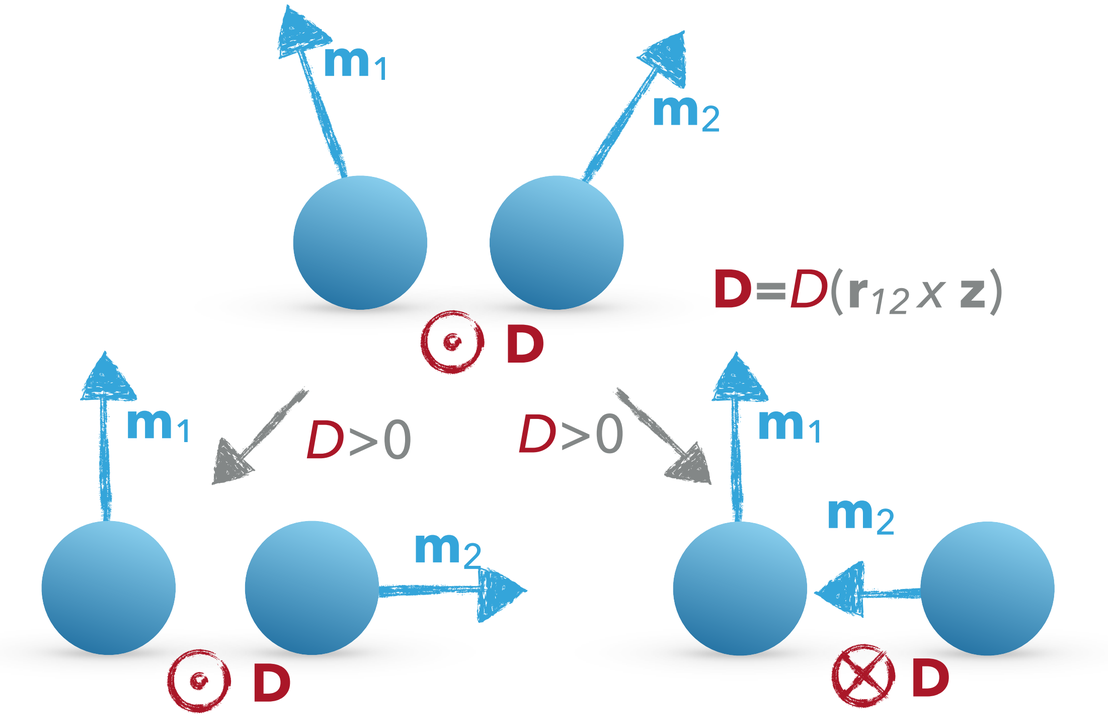
DZYALOSHINSKII–MORIYA (C_{NV})

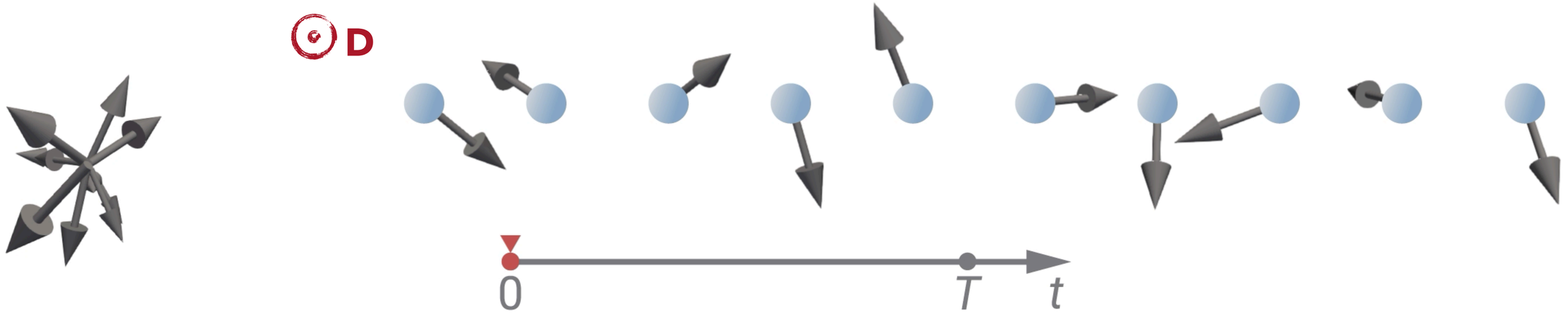
- Aligns neighbouring magnetic moments (in **m**) perpendicular to each other.
- Parameter: D (J/m²)

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

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

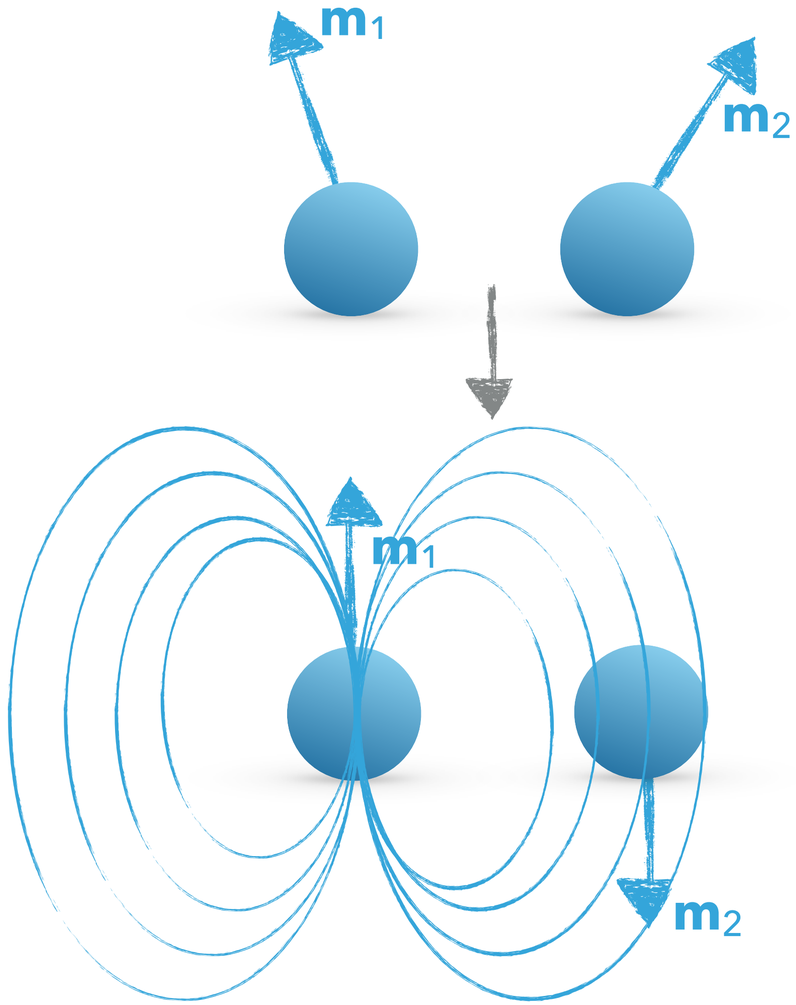
$$\left(\mathbf{D} = D(\mathbf{r}_{ij} \times \hat{\mathbf{z}})\right)$$



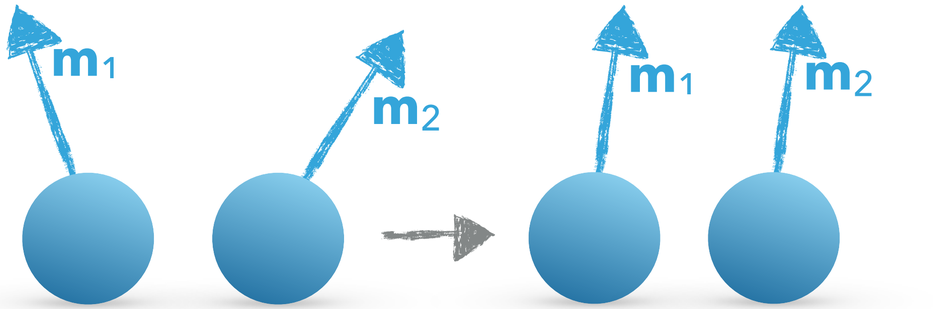


DEMAGNETISATION

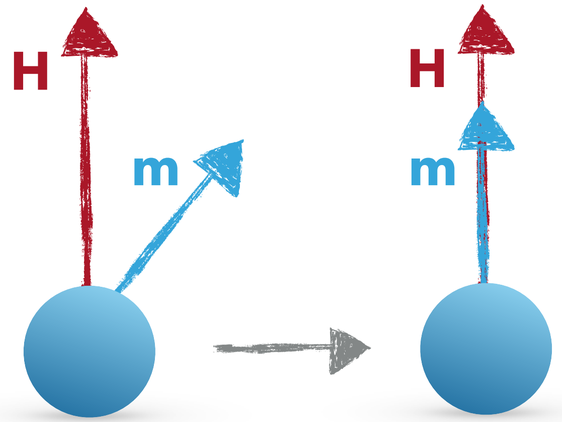
- **Long range interaction**
- It is computed from **m** and no parameter is required.
- Each magnetic moment (in **m**) “feels” the total magnetic field of all surrounding moments.
- **Computationally expensive**.
- More details in the next sessions.



EXCHANGE AND ZEEMAN

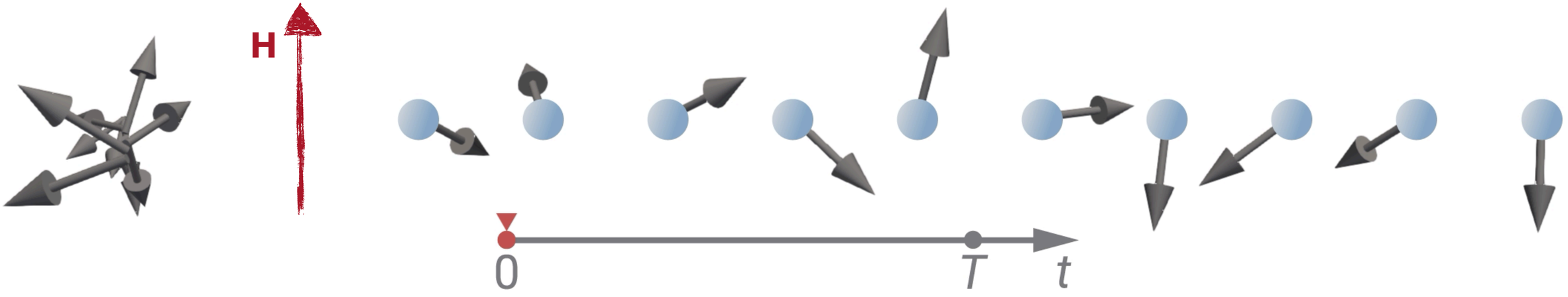


All magnetic moments parallel to each other? ✓

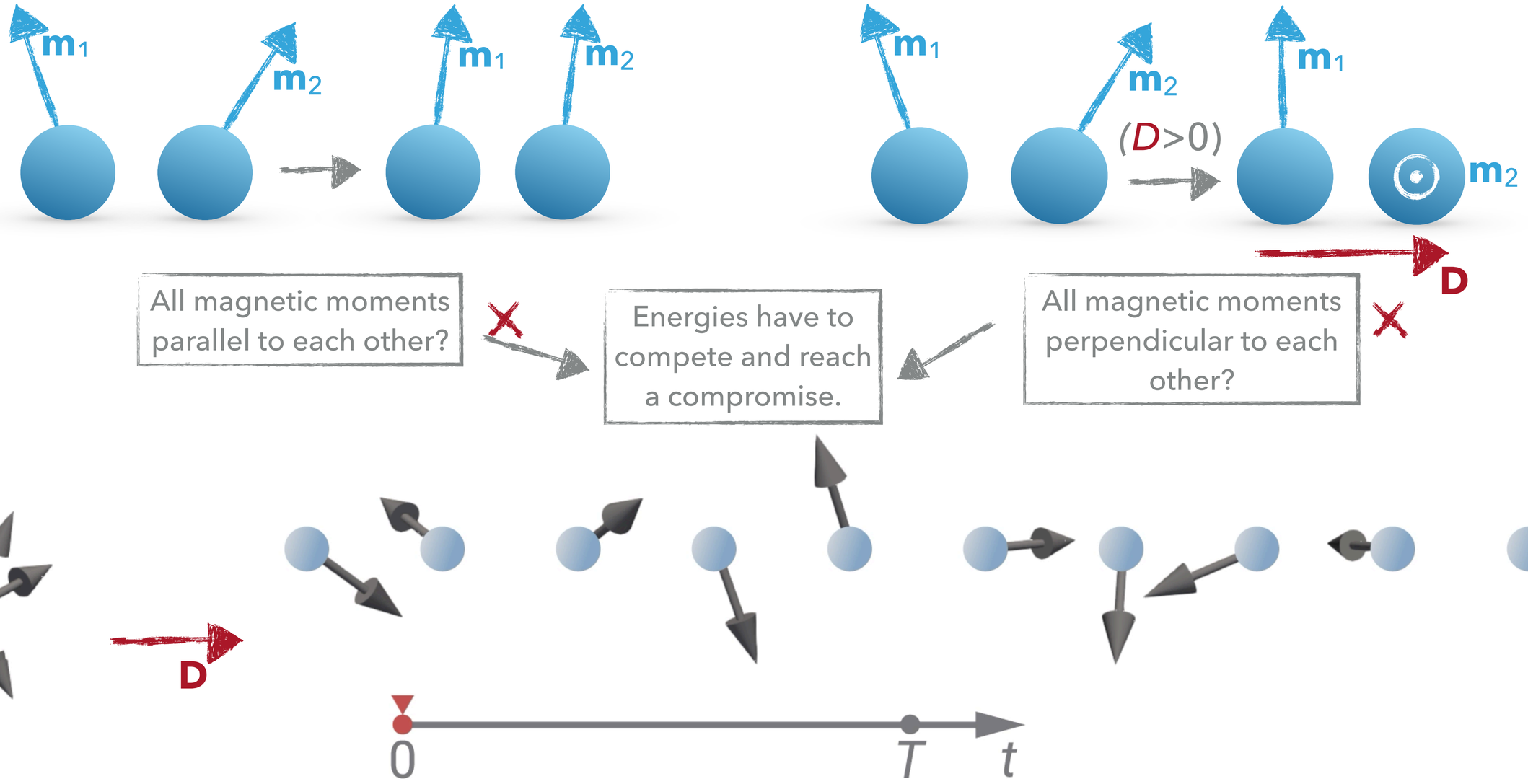


All magnetic moments parallel to **H**? ✓

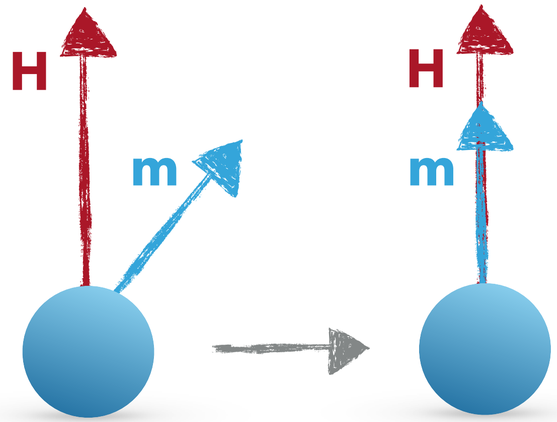
No compromise is needed.



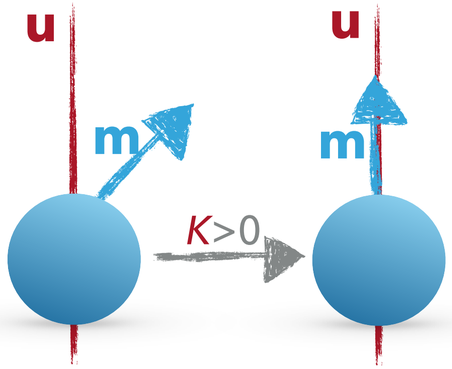
EXCHANGE AND DMI (T)

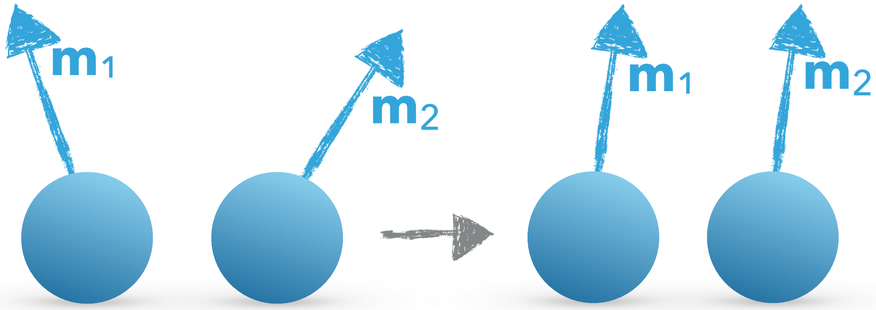


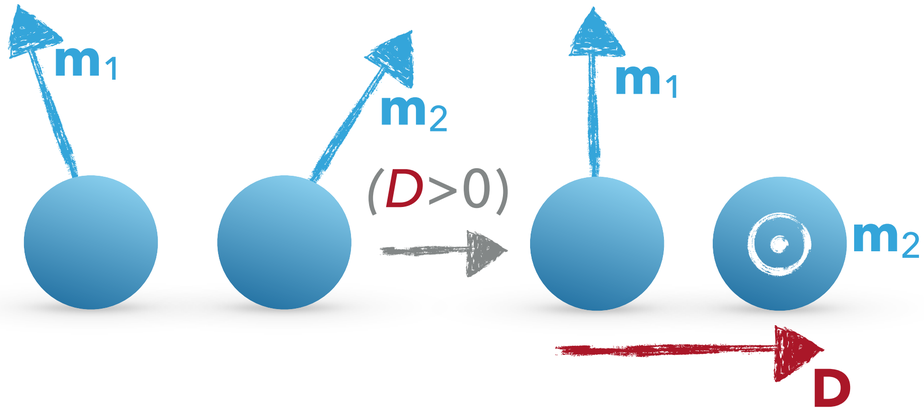
MORE COMPLICATED CASE IN A 2D SAMPLE (1/2)



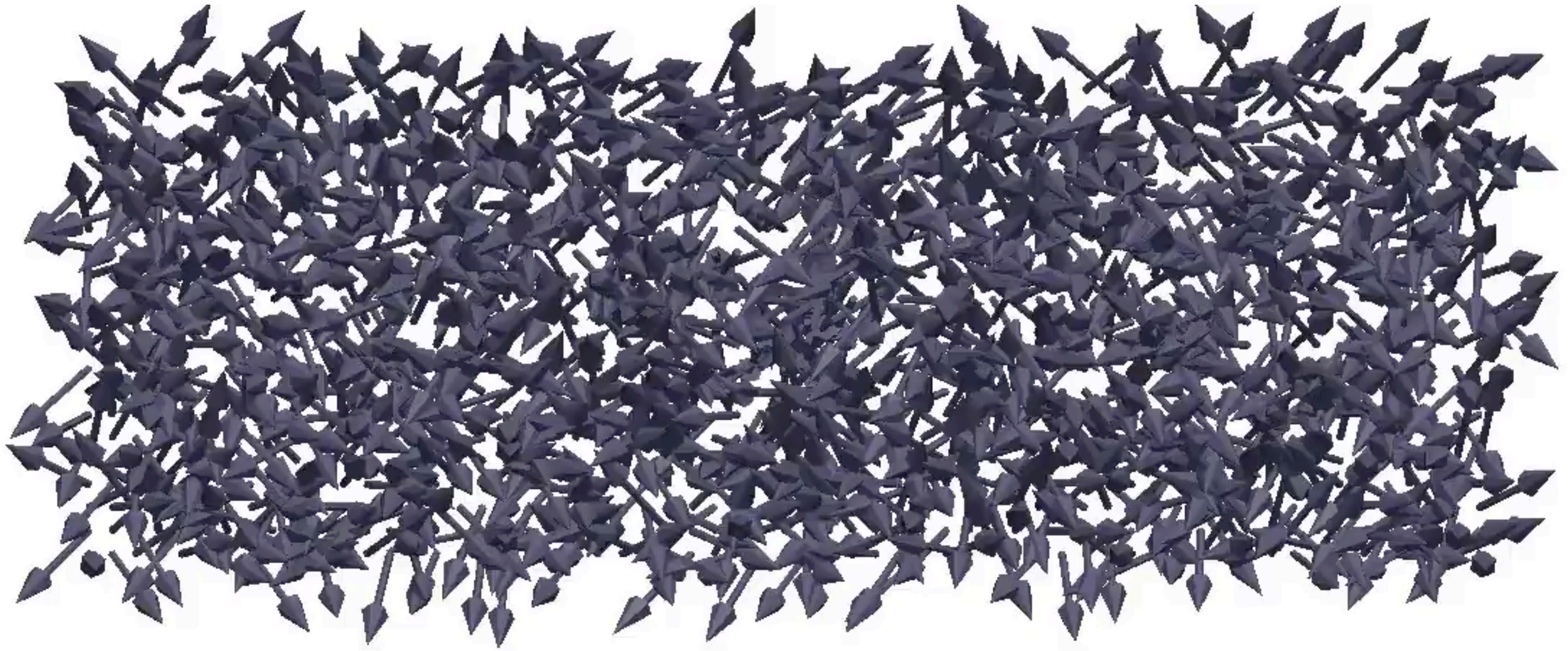
Energies have to compete and reach a compromise.







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



DYNAMICS EQUATION

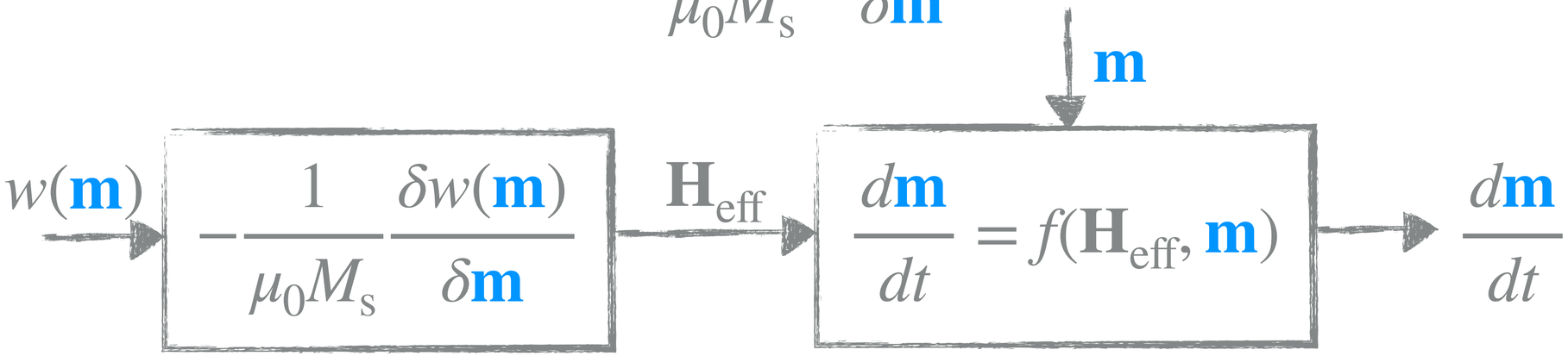
- ...tells us how magnetisation **m** wants to change in order to minimise its energy.

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

user-defined

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

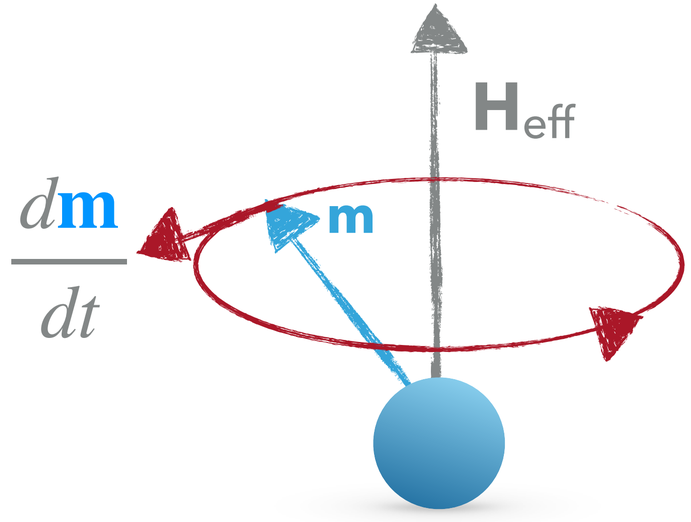
$$\mathbf{H}_{\text{eff}} = -\frac{1}{\mu_0 M_s}\frac{\delta w(\mathbf{m})}{\delta \mathbf{m}}$$



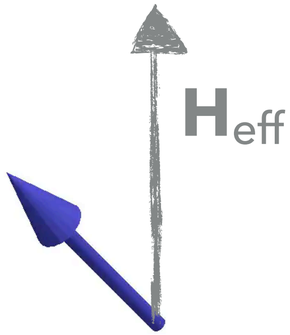
PRECESSION

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

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



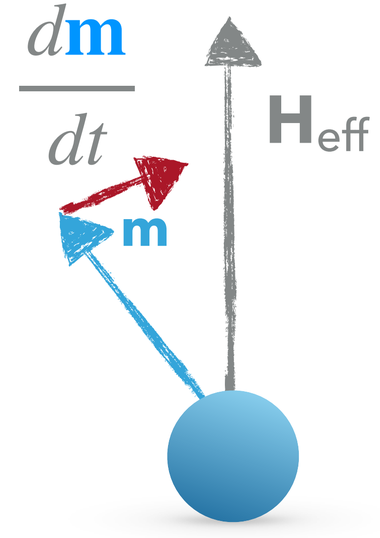
Parameter in Ubermag is γ_0, not γ.

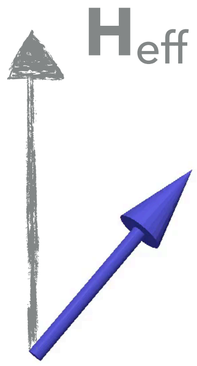


GILBERT DAMPING

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

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

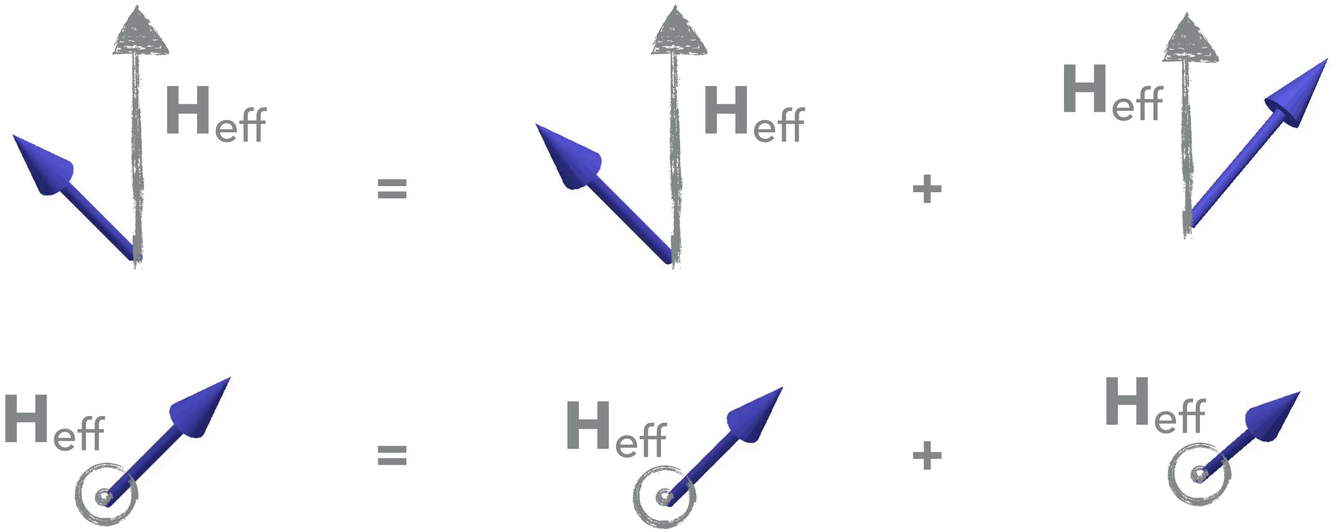




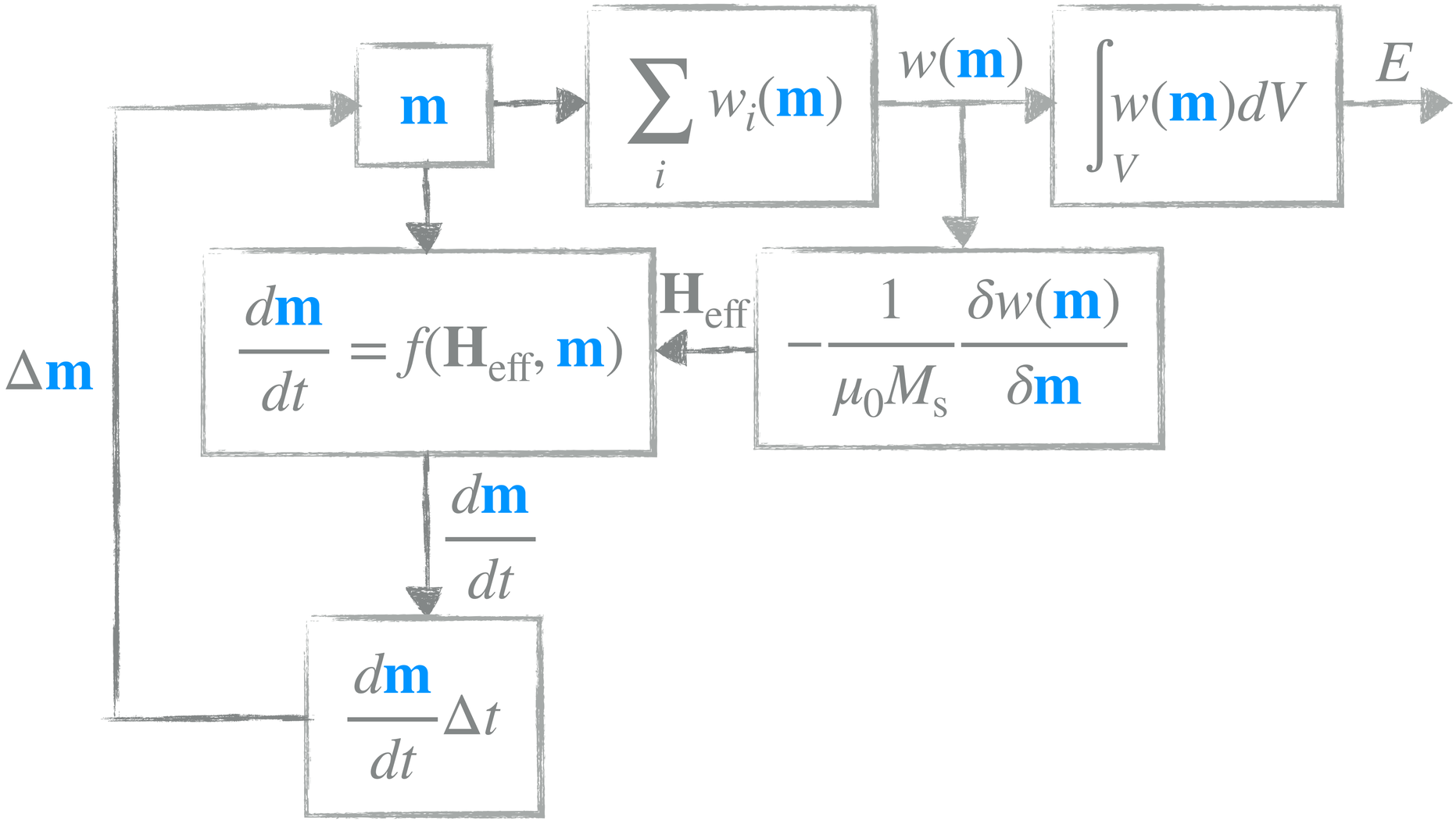
LANDAU-LIFSCHITZ-GILBERT EQUATION

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

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



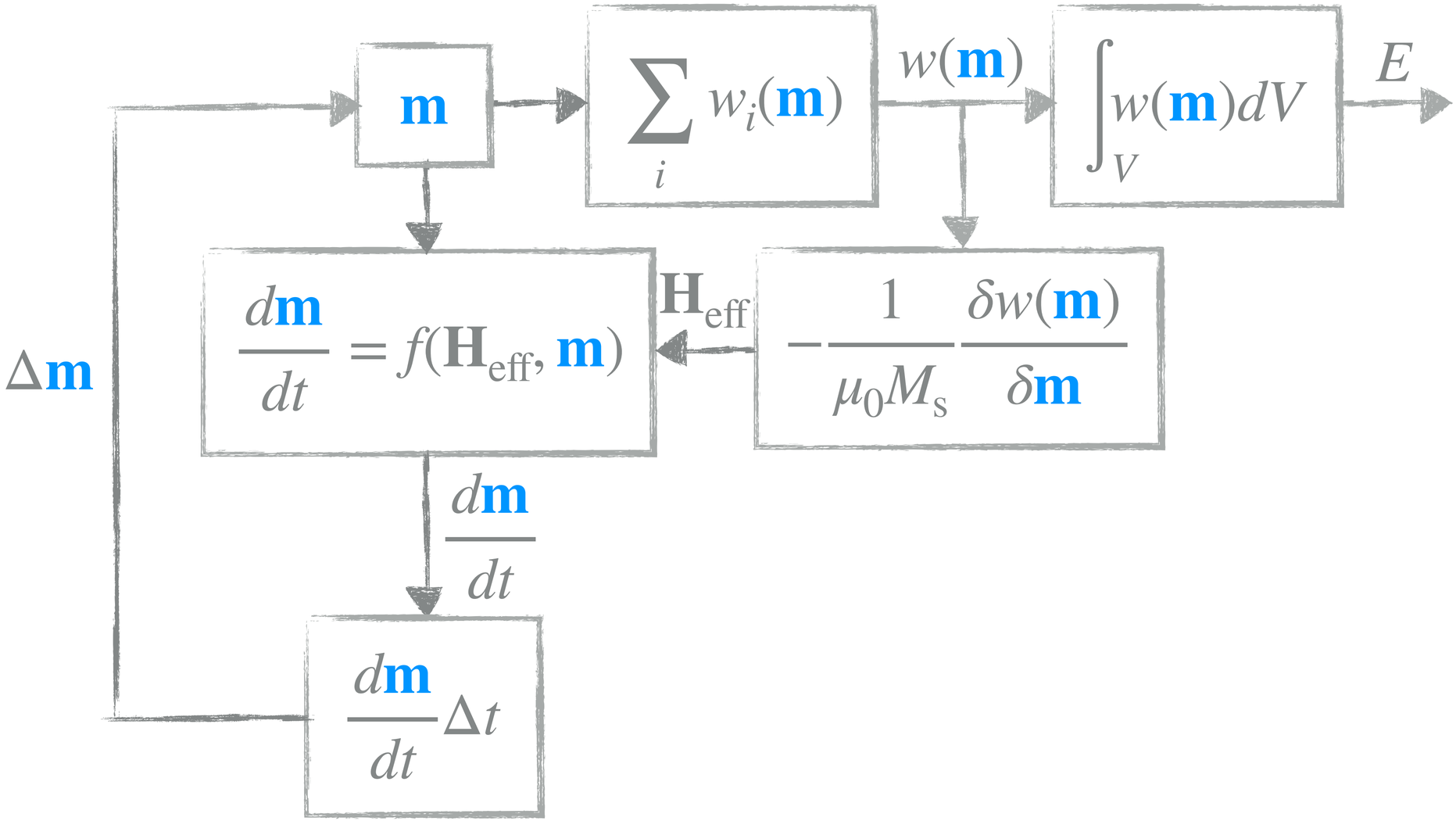
(OVERSIMPLIFIED) MICROMAGNETIC SIMULATOR



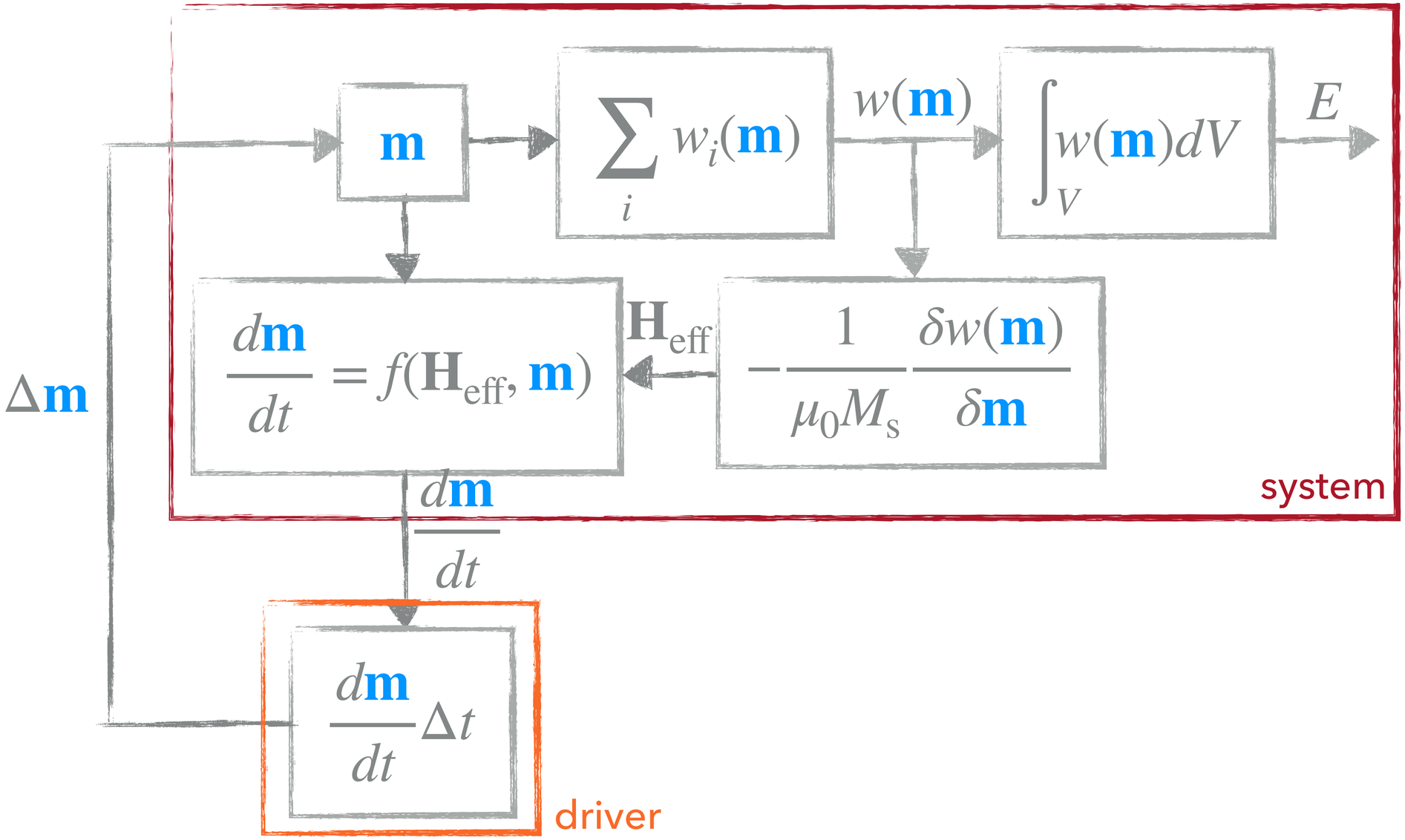
PART 2

FIRST UBERMAG SIMULATION

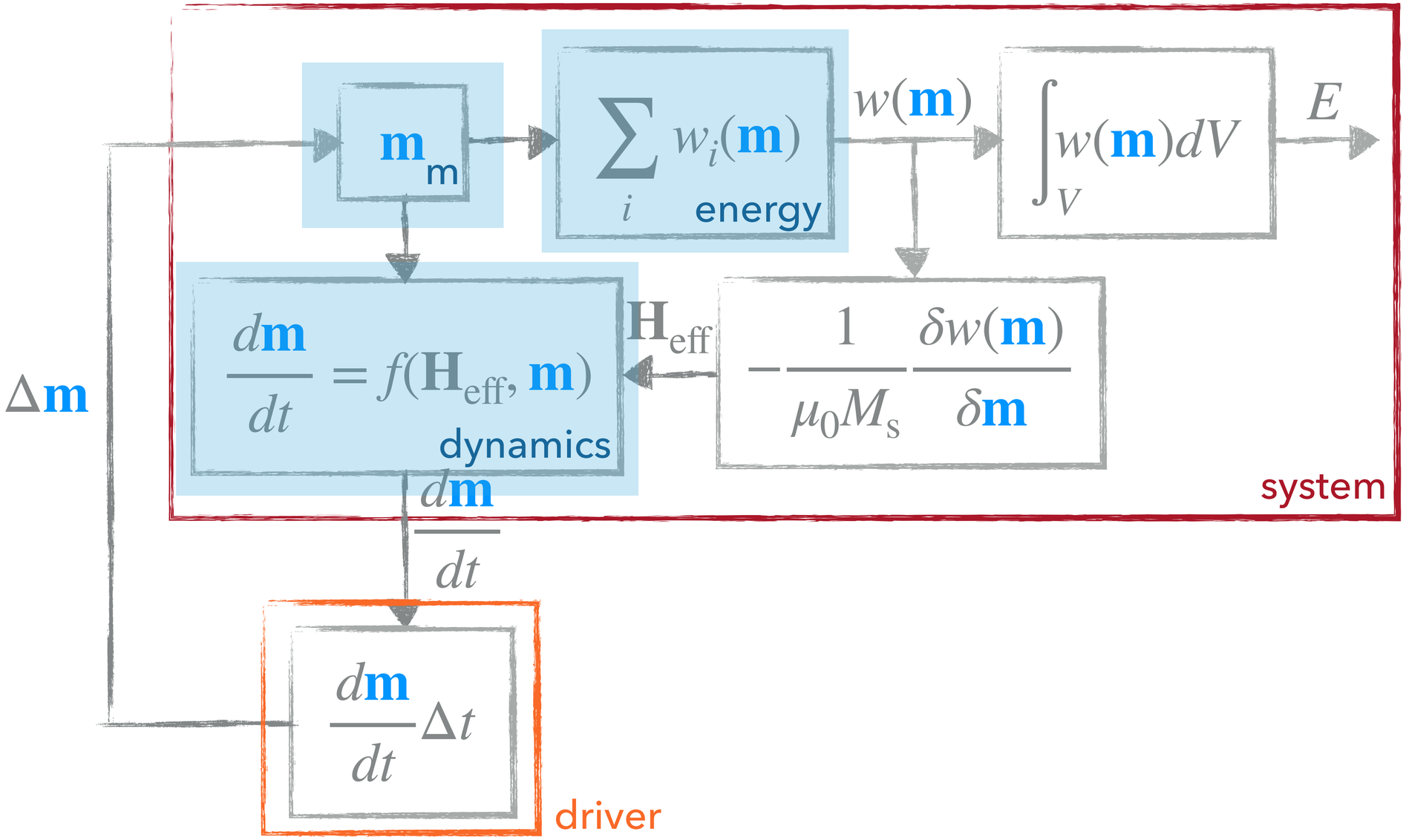
(OVERSIMPLIFIED) MICROMAGNETIC SIMULATOR



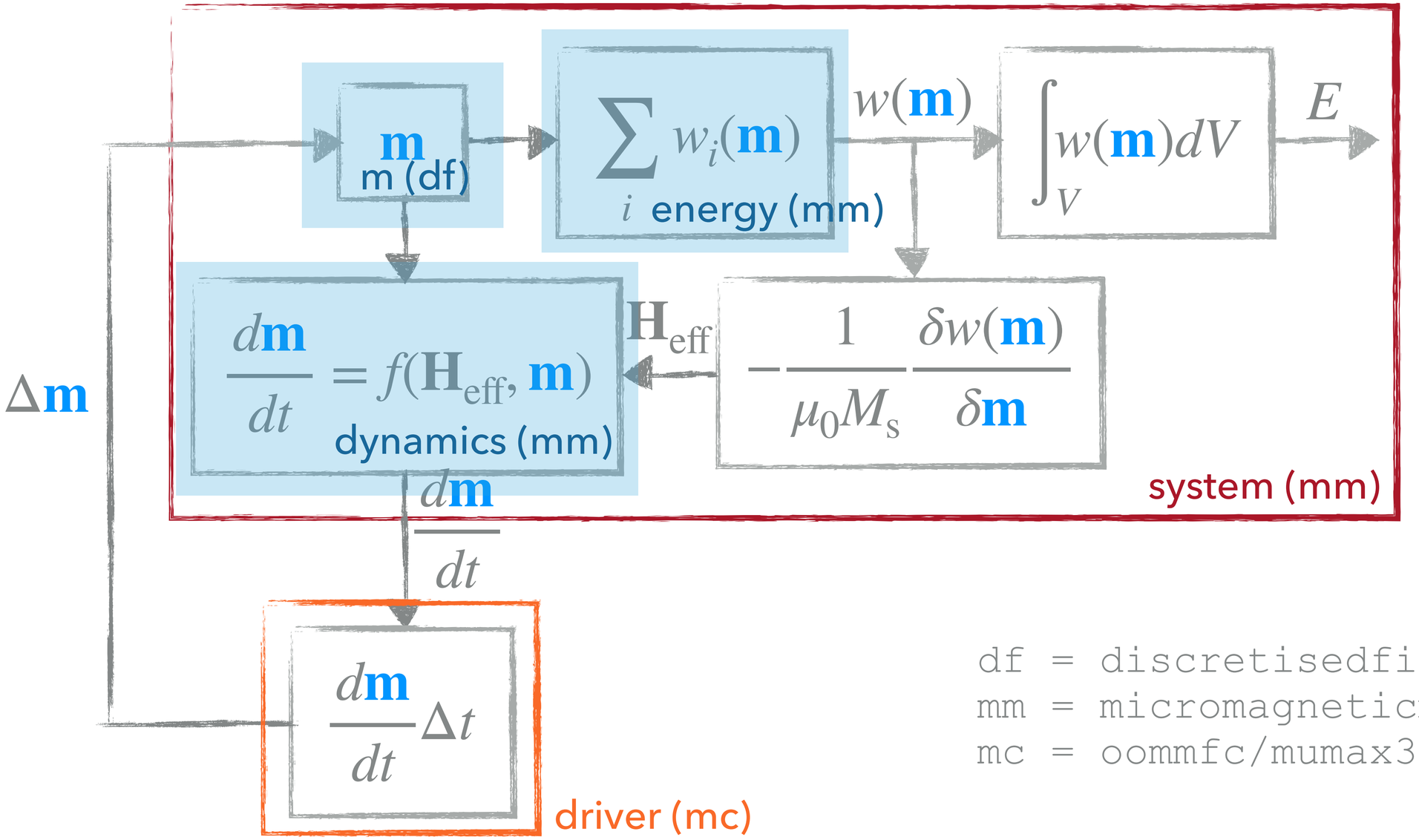
(OVERSIMPLIFIED) MICROMAGNETIC SIMULATOR



(OVERSIMPLIFIED) Mmicromagnetic simulator



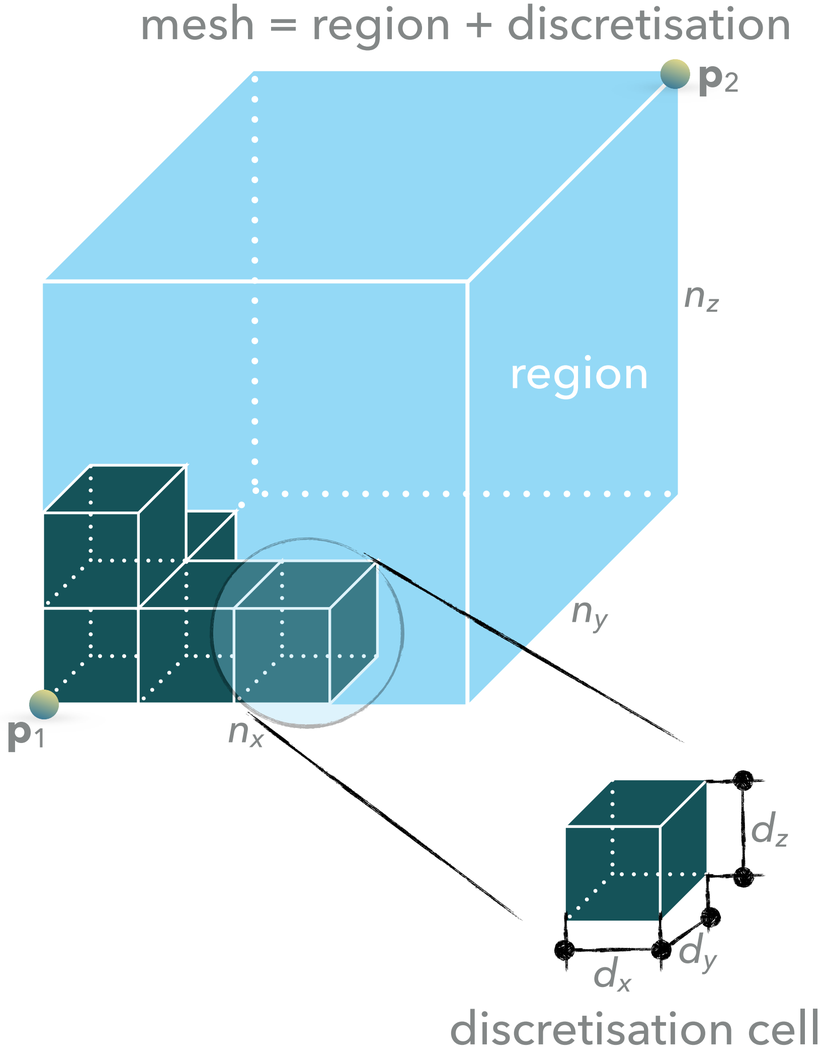
(OVERSIMPLIFIED) MICROMAGNETIC SIMULATOR




```
df = discretisedfield
mm = micromagneticmodel
mc = oommfc/mumax3c
```

DEFINING MAGNETISATION FIELD 1/2

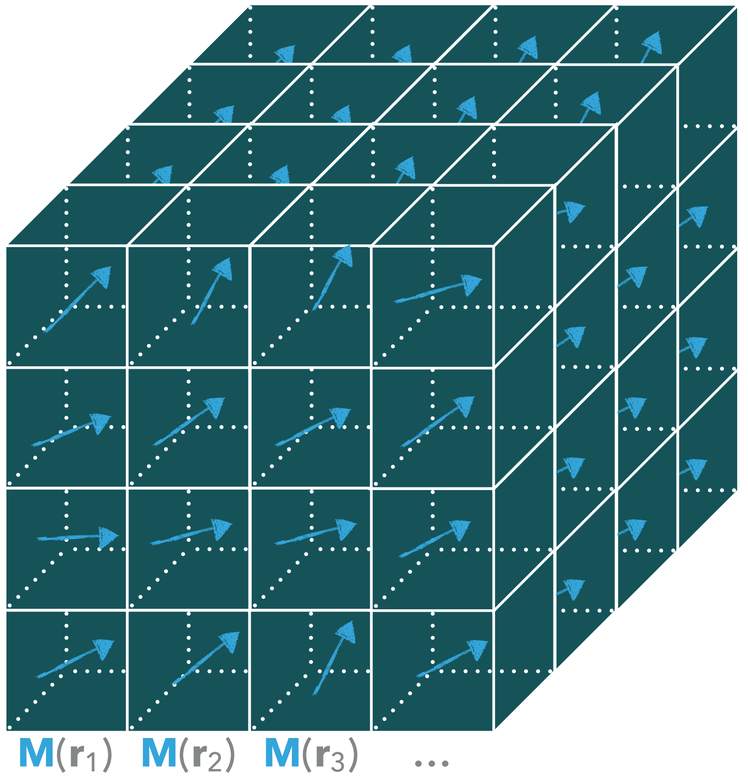
- Ubermag package: `discretisedfield`
  - `import discretisedfield as df`
- **Region** defines the domain in three-dimensional space.
  - `region = df.Region(p1=(…), p2=(…))`
- **Mesh** discretises the region
  - Either discretisation **cell size or the number of cells** can be defined.
  - `mesh = df.Mesh(region=region,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

discretised field at time t_i



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	**H** (A/m)	`mm.Zeeman`	parameter is **H** not **B**
uniaxial anisotropy	K (J/m^3), $\mathbf{u}$	`mm.UniaxialAnisotropy`	sign of K, $\lvert\mathbf{u}\rvert=1$
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, $\lvert\mathbf{u}_i\rvert=1$
demagnetisation	None	`mm.Demag`	None

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 **m** x $\mathbf{H}_{eff}$ = 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)`