Advanced VAMPIRE WORKSHOP 2017

Workshop Programme

Today	Spin models and VAMPIRE overview
Tuesday	Constrained Monte Carlo
Wednesday	Complex interactions and spin states
Thursday	Complex crystal structures and domain walls
	Roy Chantrell Talk and Dinner
Friday	Accelerated and parallel spin dynamics
	Discussion

UNIVERSITY of York

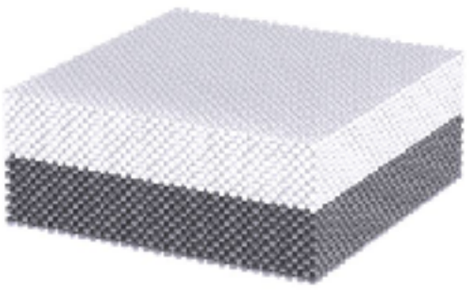
VAMPIRE features covered

Methods	Spin dynamics, Monte Carlo, Constrained and Hybrid Constrained Monte Carlo, Dipole fields
Calculations	Curie/Néel Temperature, Hysteresis loops, Temperature dependent anisotropy, Domain Wall width, ground state spin structures
Systems	Multilayers, multiple sublattices, nanoparticles, Skyrmions, antiferromagnets, complex crystal structures
Technical	Parallel and GPU acceleration, visualization and data analysis

Day 1 Overview

Brief review of atomistic spin models

VAMPIRE code overview

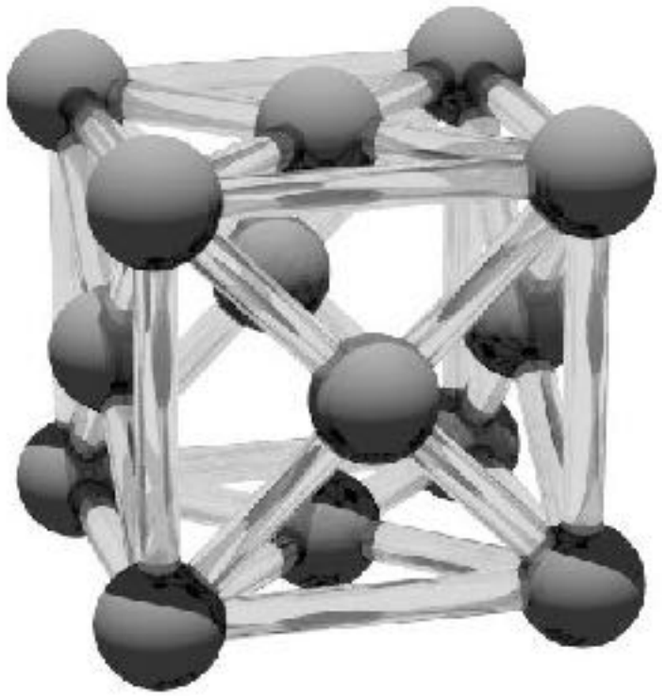


Practicals

UNIVERSITY of York

richard.evans@york.ac.uk

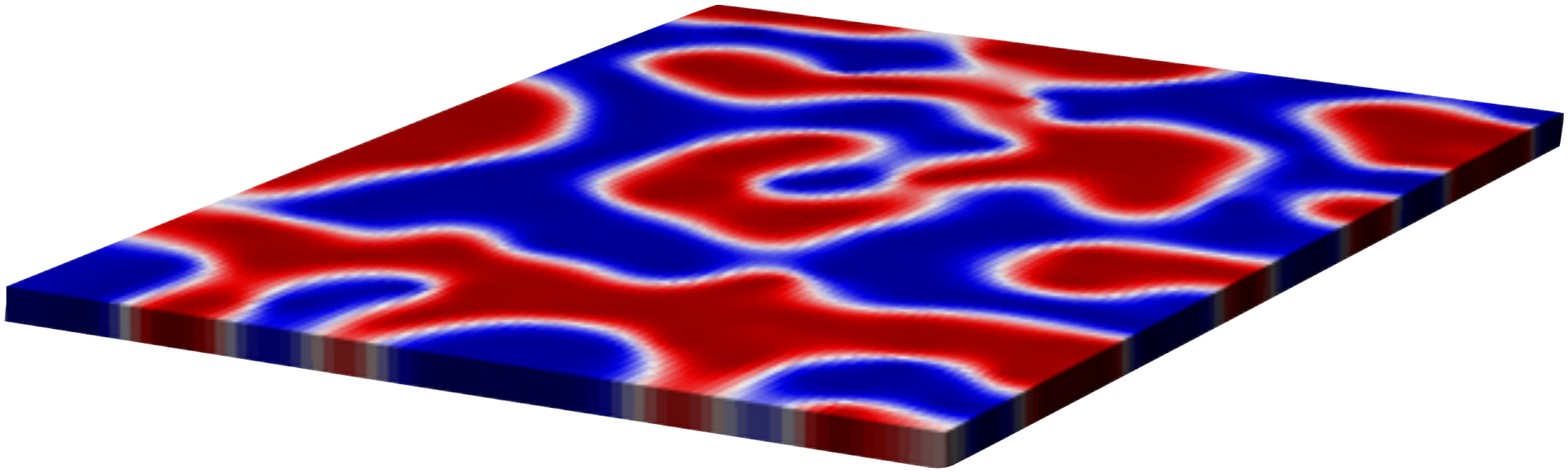
Bulk magnetic properties intrinsically tied to electronic structure



Magnetic moments
Exchange interactions
Anisotropy

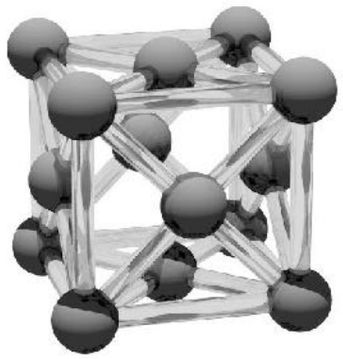
UNIVERSITY of York

Numerical micromagnetics enables us to really understand how magnetic materials behave

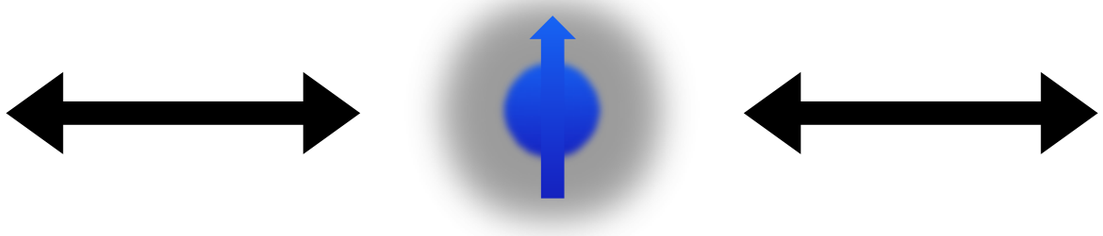


UNIVERSITY of York

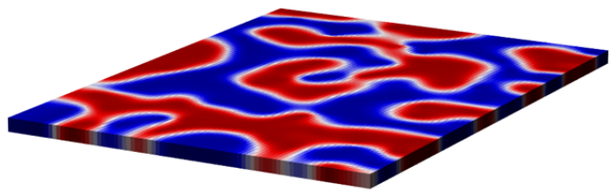
Natural link between length scales



Ab-initio



Atomistic



Micromagnetics

UNIVERSITY of York

Atomistic spin models

The ‘spin’ Hamiltonian

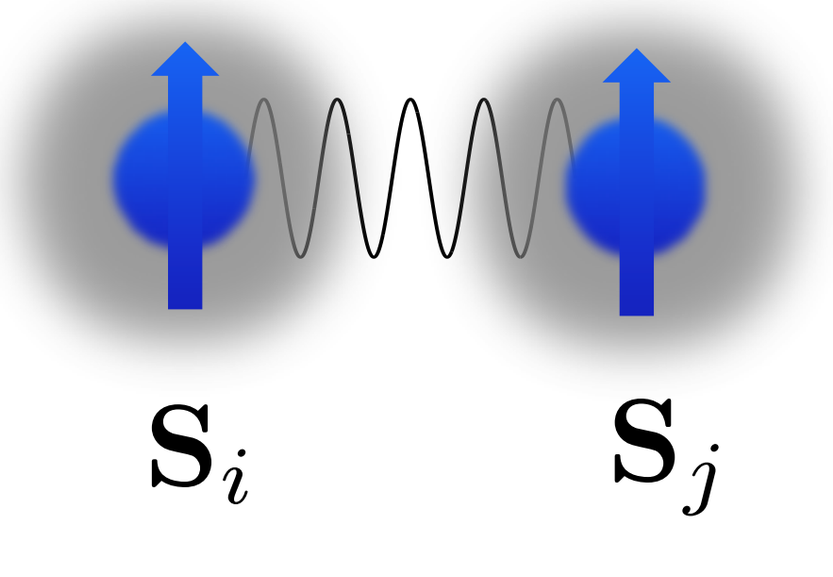
$$\mathcal{H} = \mathcal{H}_{\mathrm{exc}} + \mathcal{H}_{\mathrm{ani}} + \mathcal{H}_{\mathrm{app}}$$

Exchange

Anisotropy

Applied Field

Foundation of the atomistic model is Heisenberg exchange

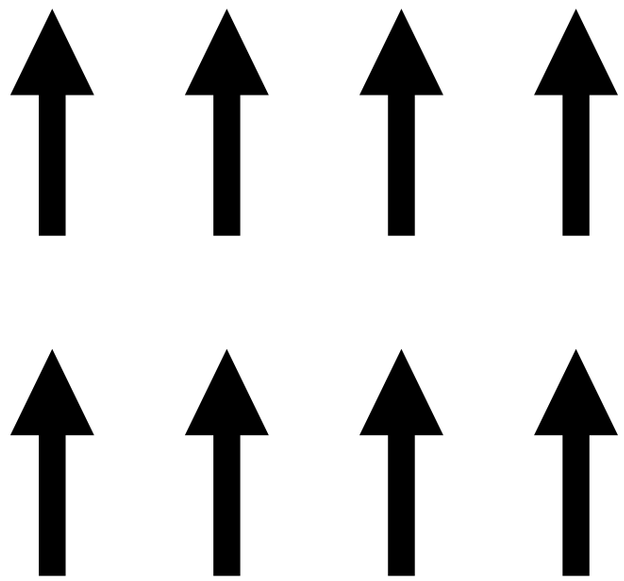


$$\mathcal{H}_{\text{exc}} = -\sum_{i<j} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j$$

Natural discrete limit of magnetization

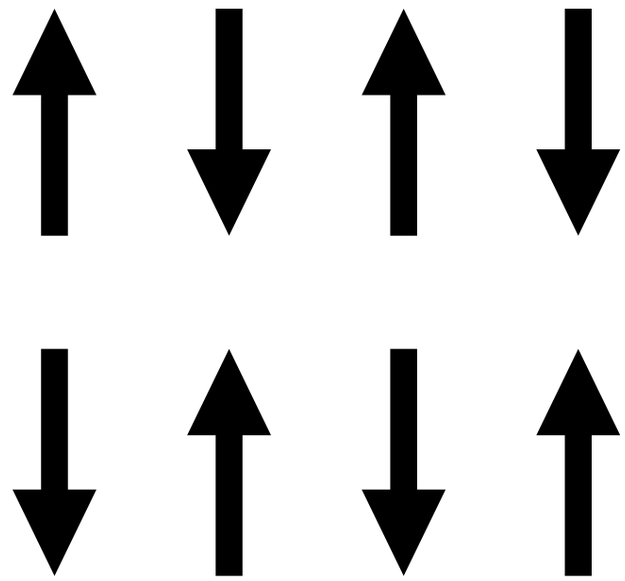
UNIVERSITY of York

Exchange interaction determines type of magnetic ordering



$J_{ij} > 0$

Ferromagnet



$J_{ij} < 0$

Anti-ferromagnet

Exchange energy defines the Curie / Néel temperature of the material

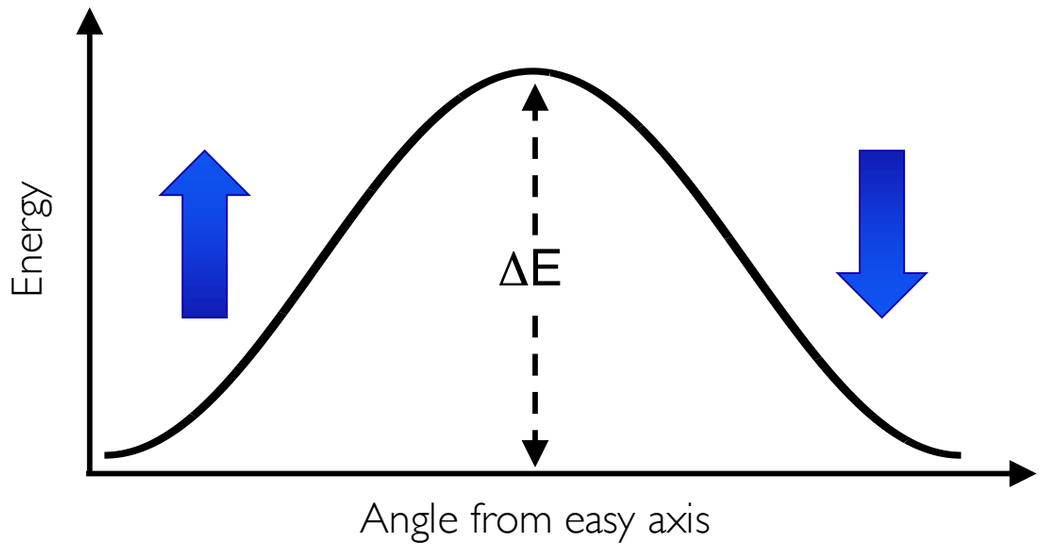
$$J_{ij} = \frac{3k_B T_\mathrm{c}}{\epsilon z}$$

Mean field approximation with correction factor for spin waves

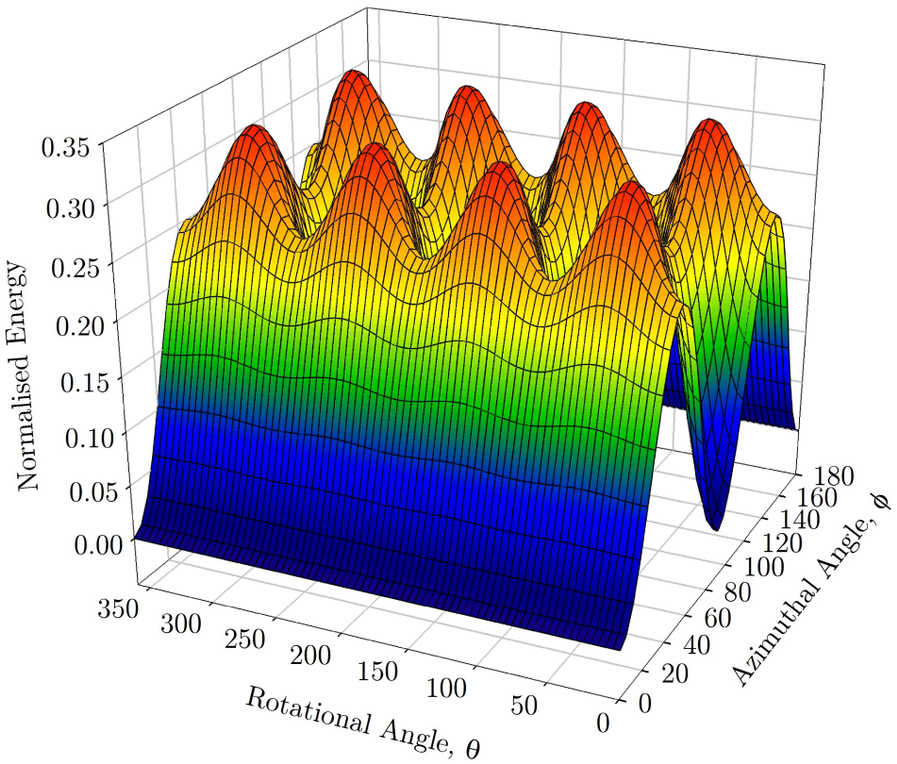
D. A. Garanin, Physical Review B **53**, 11593 (1996)

Magnetic anisotropy energy

Uniaxial



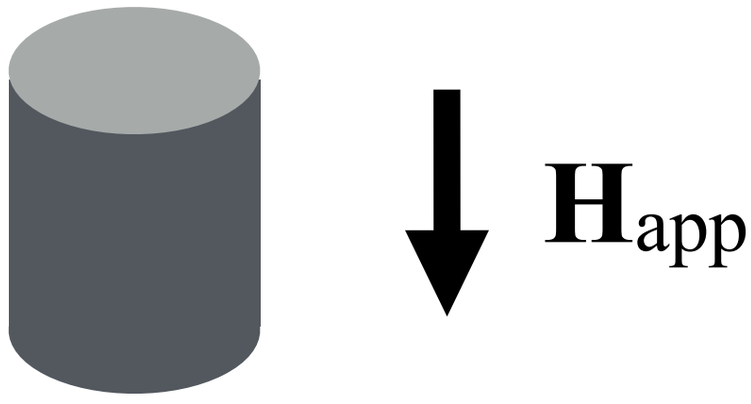
Cubic



$$\mathcal{H}_{\text{ani}}^{\text{uni}} = -k_{\text{u}} \sum_i \left(\mathbf{S}_i \cdot \boldsymbol{e}\right)^2$$

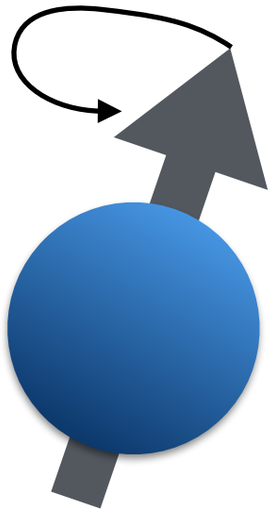
$$\mathcal{H}_{\text{ani}}^{\text{cub}} = \frac{k_{\text{c}}}{2} \sum_i \left(S_x^4 + S_y^4 + S_z^4\right)$$

Externally applied fields



$$\mathcal{H}_{\text{app}} = -\sum_i \mu_{\text{s}} \mathbf{S}_i \cdot \mathbf{H}_{\text{app}}$$

Integration methods



Ising model

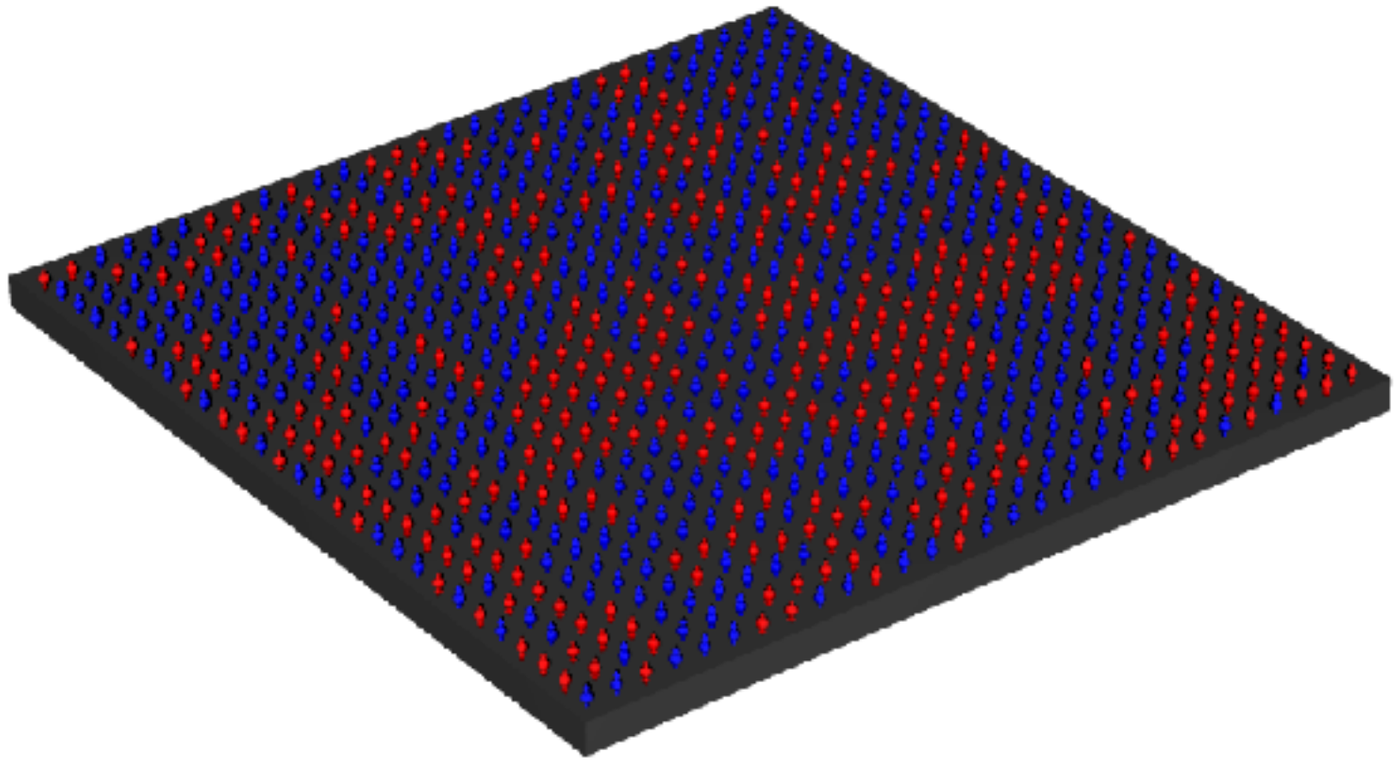
Beitrag zur Theorie des Ferromagnetismus [1]).

Von **Ernst Ising** in Hamburg.

(Eingegangen am 9. Dezember 1924.)

Es wird im wesentlichen das thermische Verhalten eines linearen, aus Elementarmagneten bestehenden Körpers untersucht, wobei im Gegensatz zur Weissschen Theorie des Ferromagnetismus kein molekulares Feld, sondern nur eine (nicht magnetische) Wechselwirkung benachbarter Elementarmagnete angenommen wird. Es wird gezeigt, daß ein solches Modell noch keine ferromagnetischen Eigenschaften besitzt und diese Aussage auch auf das dreidimensionale Modell ausgedehnt.

1. Annahmen. Die Erklärung, die P. Weiss [2]) für den Ferromagnetismus gegeben hat, ist zwar formal befriedigend, doch läßt sie besonders die Frage nach einer physikalischen Erklärung der Hypothese des molekularen Feldes offen. Nach dieser Theorie wirkt auf jeden

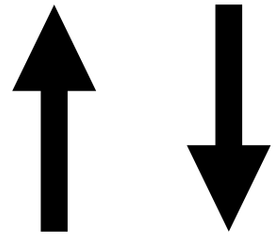


Simplest model of spin-1/2 ferromagnet phase transition

“Toy model”

Ising model

Two allowable states, up, down

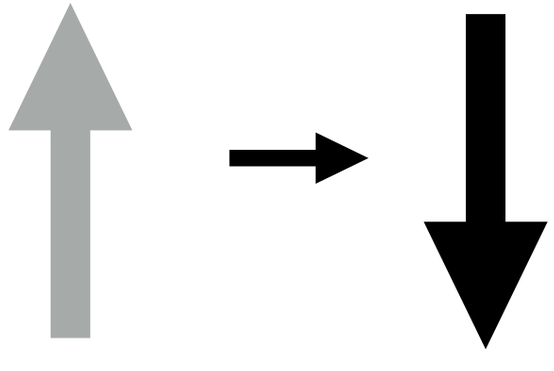


Energy barrier between states defined by exchange energy

$$\mathcal{H}_{\text{exc}} = -\sum_{i<j} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j$$

Monte Carlo algorithm

1. Pick a new trial state (or move)



2. Evaluate energy before (E_1) and after (E_2) spin flip

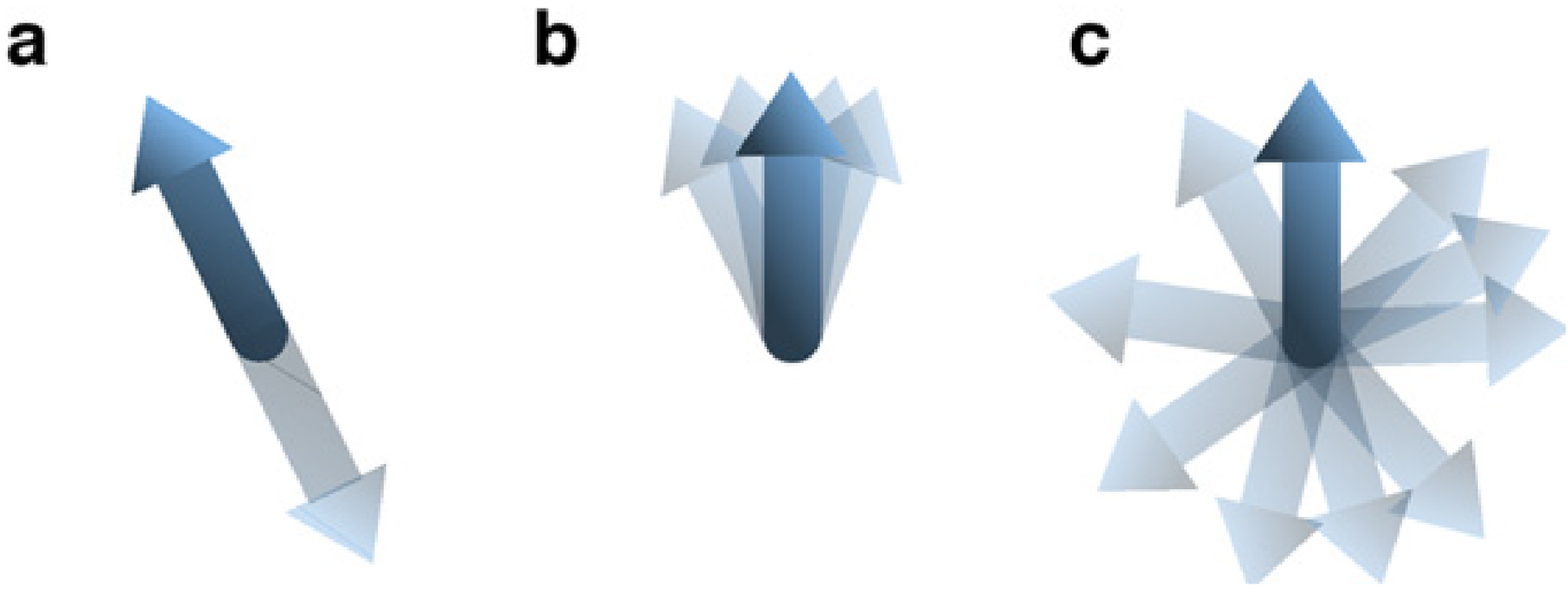
3. Evaluate energy difference between states

$$\Delta E = (E_2 - E_1)$$

4. Accept move with probability

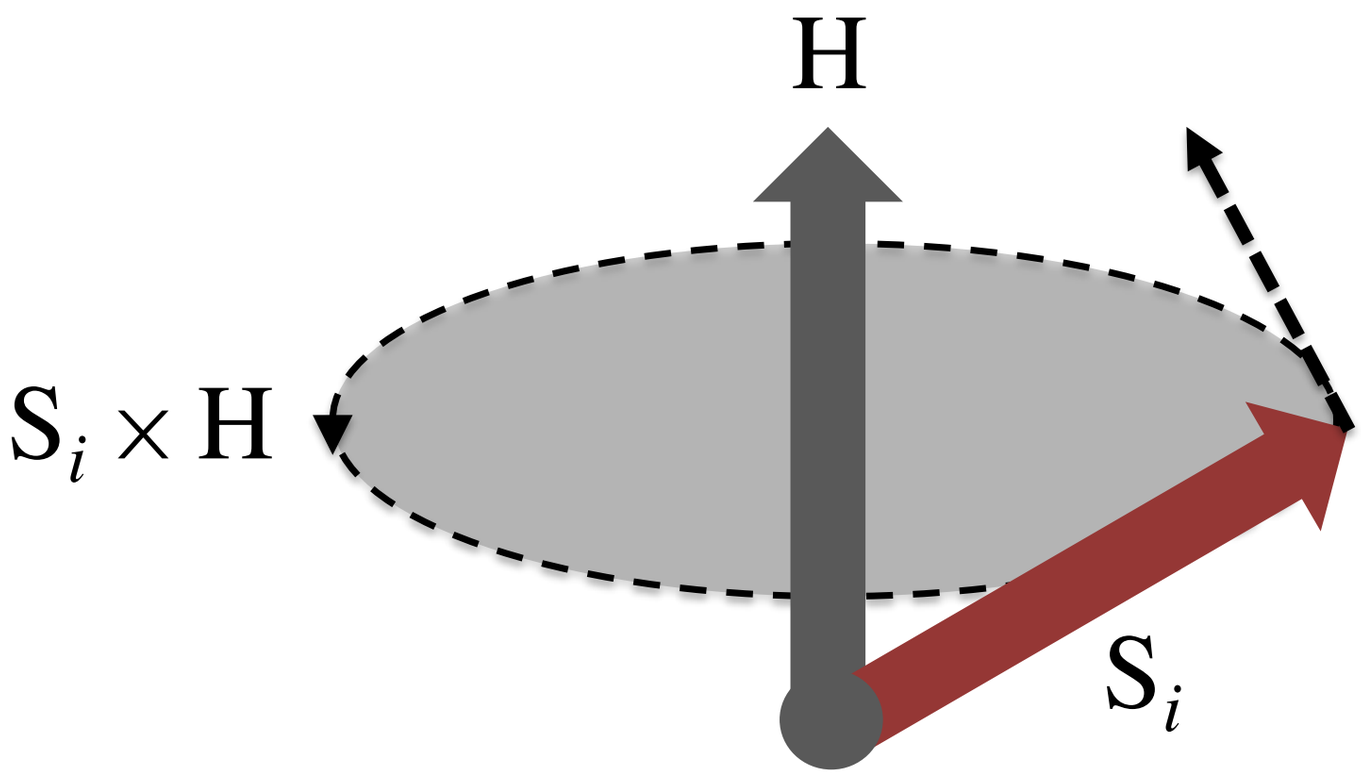
$$\exp(-\Delta E/k_B T)$$

Extension to 3D Heisenberg model straightforward



Use a combination of different trial moves

Atomistic Spin dynamics
(Landau-Lifshitz-Gilbert equation)

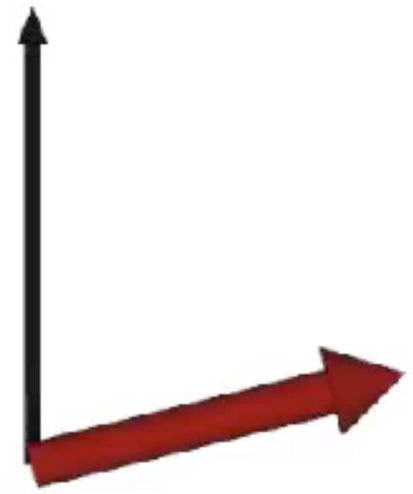


$$\frac{\partial \mathbf{S}_i}{\partial t} = -\frac{\gamma}{(1+\lambda^2)}[\mathbf{S}_i \times \mathbf{H}^i_{\text{eff}} + \lambda \mathbf{S}_i \times (\mathbf{S}_i \times \mathbf{H}^i_{\text{eff}})]$$

UNIVERSITY of York

Stochastic Landau-Lifshitz-Gilbert equation

$$\mathbf{H}_{\mathrm{eff}}^{i} = -\frac{1}{\mu_{\mathrm{s}}}\frac{\partial \mathcal{H}}{\partial \mathbf{S}_i} + \mathbf{H}_{\mathrm{th}}^{i,\delta}.$$

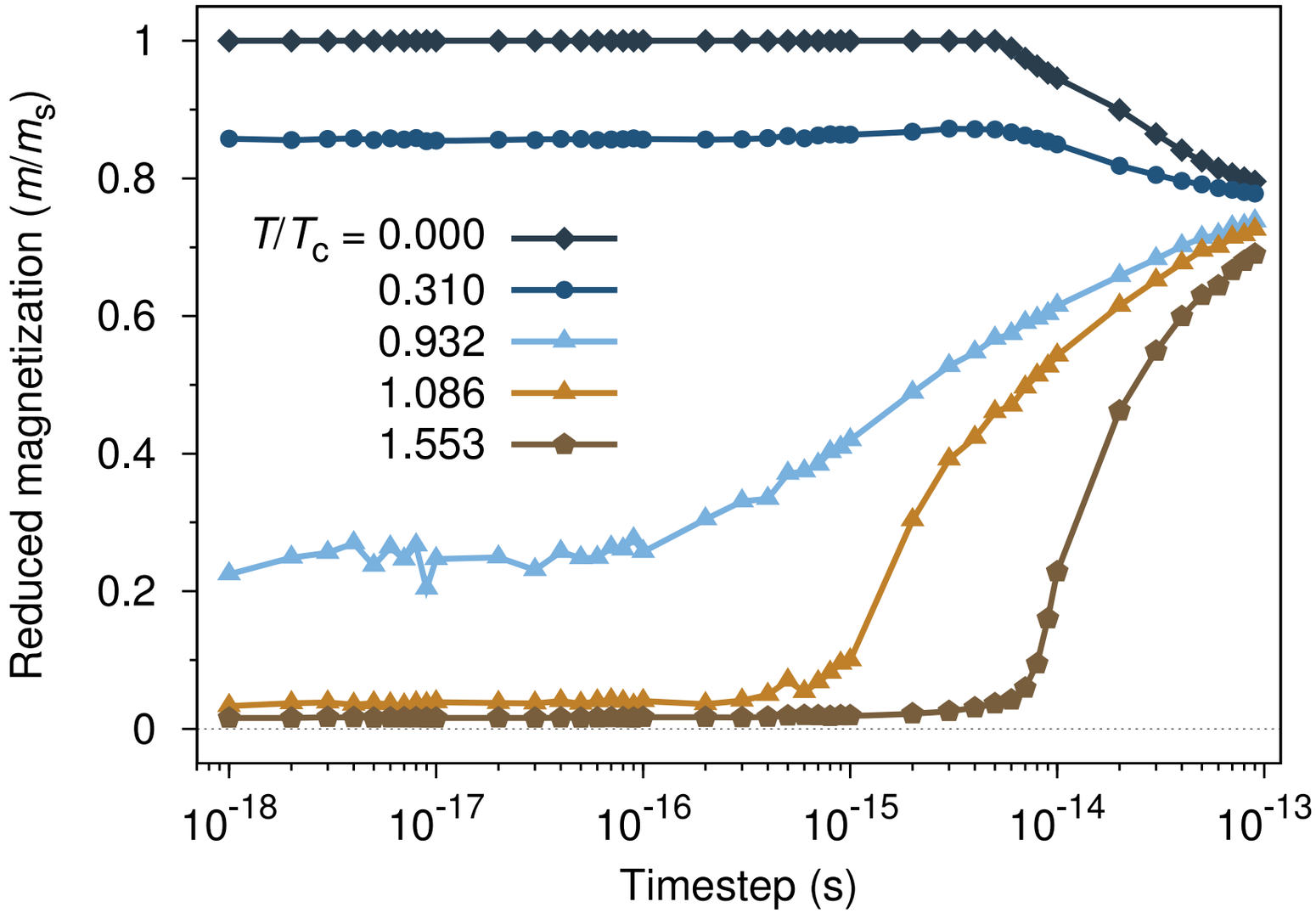


$$\mathbf{H}_{\mathrm{th}}^{i} = \mathbf{\Gamma}(t)\sqrt{\frac{2\lambda k_{\mathrm{B}} T}{\gamma \mu_{\mathrm{s}} \Delta t}}$$

Spin dynamics is the magnetic analogue of molecular dynamics

UNIVERSITY of York

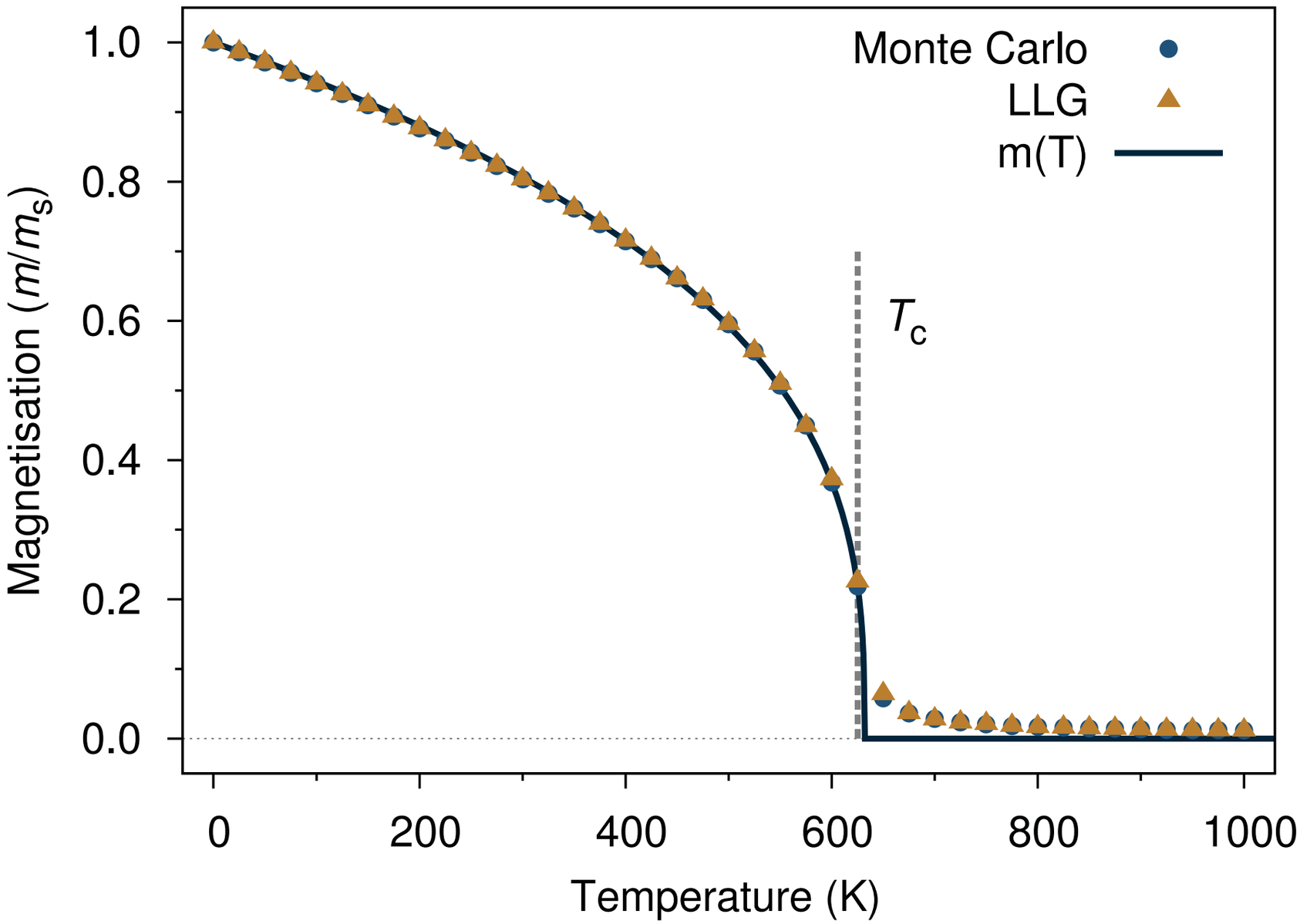
Timesteps with the LLG equation



Generally always need 0.1 fs (10^{-16} s) time steps

1 fs (10^{-15} s) maybe OK for simple ferromagnets, $T << T_c$

Comparison of LLG and Monte Carlo



Use Monte Carlo for equilibrium properties

Use LLG for dynamic processes

VAMPIRE

Code Overview

Programming language and approach

Written in C++ (2011 standard)

A mixture of object oriented (creation routines) and functional (high performance) programming styles

Supports Message Passing Interface (MPI) parallelisization, CUDA and OpenCL in alpha test.

Increasingly modular code base, work in progress

Version control

Managed with git version control system and hosted at Github

Open source with mixture of GPL and BSD licenses

Branches maintain different parallel versions of the code

- master branch - official releases (very old)
- develop - current up to date version (a bug or two)
- cuda - testing branch for CUDA version

New module structure for new additions

src/module/data.cpp | variables and arrays in module

src/module/interface.cpp | user interface to module

src/module/initialise.cpp | function to initialize module data and variables

src/module/internal.h | header file for sharing variables within a module

hdr/module.h | interface to main VAMPIRE code

Modules

Each module is self contained and only interacts with the main vampire code with a defined interface in the module header file

Each module has its own namespace to separate it from the main code

Not all code is in modules - but work is underway

VAMPIRE execution flowchart

input
material.mat
unit cell.ucf

Read input files and initialize system variables

create

Generate crystal structure and particle shape and initialize high performance arrays

simulate

Initialize general simulation properties and select desired program to run

program

Determine changing parameters such as time, temperature, field and integrate system

spin Hamiltonian

Calculate exchange energy, anisotropy, applied fields

fields

Calculate the local field on each spin

spin dynamics (LLG)

Routine to perform a single LLG timestep for the system

integrate

Integrate system using spin dynamics or Monte Carlo

random numbers

Calculate thermal fields with Mersenne Twister/Ziggurat algorithm

statistics

Calculate average/ instant magnetization, susceptibility, and magnetization distribution

output

Output time evolution of statistics and snapshots of the spin configuration

VAMPIRE Input files

Control of the code is through plain text files

input
Main control file specifying system parameters, program, time steps, integrators, global fields, data output

material file
Lists magnetic parameters for different atom types in the simulation and provides a way to identify different magnetic states, interactions, etc

Unit Cell File specification (.ucf)

```
# Unit cell size:
3.854  3.854  3.715
# Unit cell vectors:
1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
# Atoms num, id cx cy cz mat lc hc
2
0   0.0  0.0   0   0   0   0
1   0.5  0.5   0   0   1   0
# Interactions n exctype, id i j dx dy dz Jij
2718   tensorial
0   0   0   -1  -4  -5  2.22436e-27   2.22436e-27   2.35828e-27
1   0   0   0   -4  -5  4.44872e-27   4.44872e-27   4.71655e-27
2   0   0   1   -4  -5  2.22436e-27   2.22436e-27   2.35828e-27
...
```

Contains atomic structure and interactions but with no restrictions for symmetry or number of interactions

Practical 1: compiling the code

Getting the source code

Download the code from github

```
git clone https://github.com/richard-evans/vampire/
```

Checkout the develop branch

```
git checkout develop
```

Compiling the code

Different make targets for different compilers:

```
g++ (default)
intel (icc)
llvm (macOS)
cray/archer
```

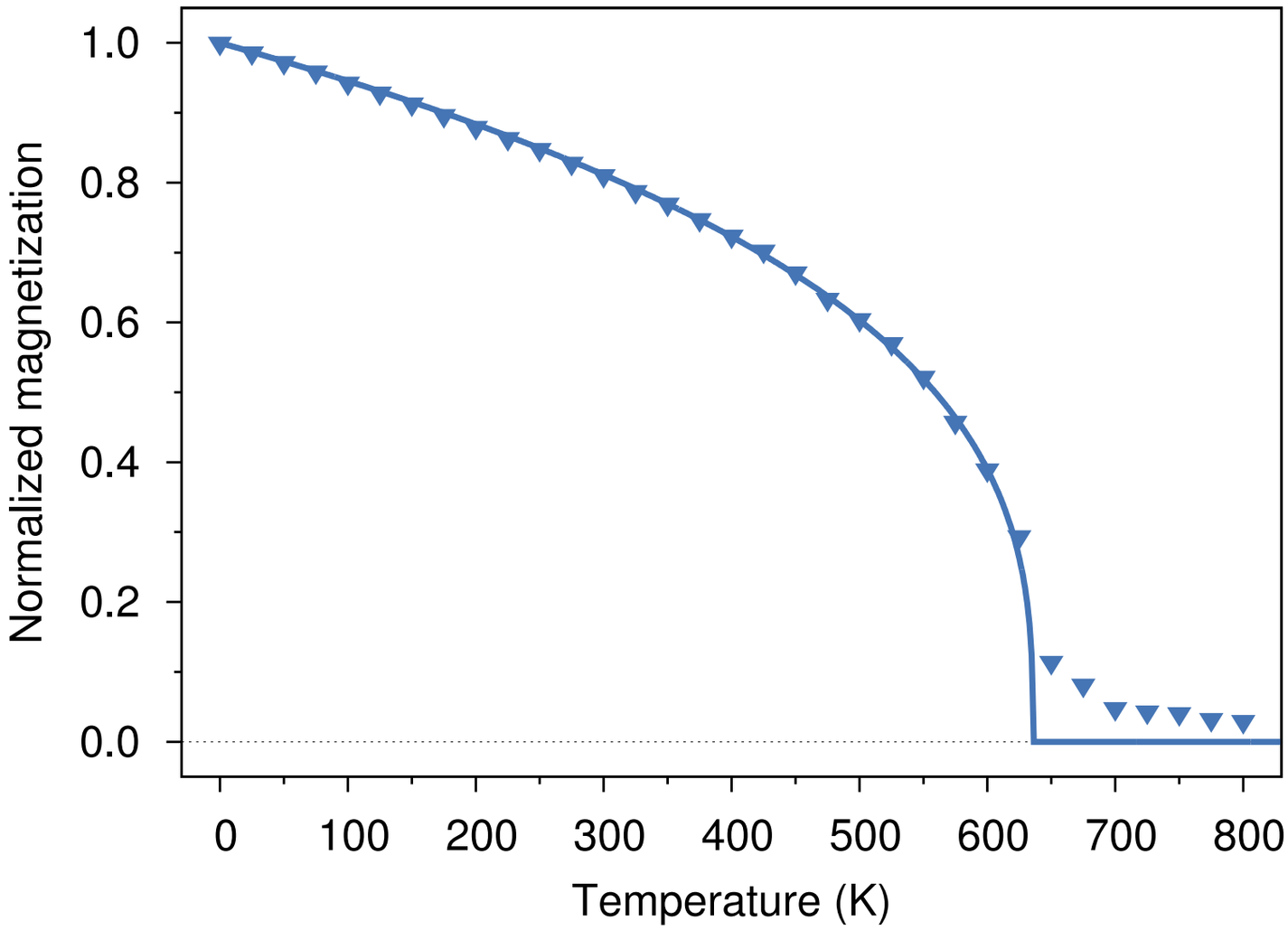
Different targets for different functionality

```
serial
parallel
gcc-cuda
```

Combine compiler and version to compile code

```
make serial
make serial-llvm
```

Practical 2: Curie temperature



Setting up a simulation in Vampire

input file
(program control)

```
#-----------------------------------------
# Creation attributes:
#-----------------------------------------
create:crystal-structure=fcc
create:periodic-boundaries-x
create:periodic-boundaries-y
create:periodic-boundaries-z
#-----------------------------------------
# System Dimensions:
#-----------------------------------------
dimensions:unit-cell-size = 3.524 !A
dimensions:system-size-x = 4.0 !nm
dimensions:system-size-y = 4.0 !nm
dimensions:system-size-z = 4.0 !nm
…
```

material file
(material properties)

```
#-------------------------------------------------
# Number of Materials
#-------------------------------------------------
material:num-materials=1
#-------------------------------------------------
# Material 1 Nickel Generic
#-------------------------------------------------
material[1]:material-name=Ni
material[1]:damping-constant=0.01
material[1]:exchange-matrix[1]=2.757e-21
material[1]:atomic-spin-moment=0.606 !muB
material[1]:uniaxial-anisotropy-constant=0.0
material[1]:material-element=Ni
```

Spin Hamiltonian for Ni

$$\mathscr{H} = -\sum_{i<j} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j - \sum_i k_{\mathrm{u}} S_{i,z}^2$$

Ni.mat

```
#-----------------------------------------------------
# Number of Materials
#-----------------------------------------------------
material:num-materials=1
#-----------------------------------------------------
# Material 1 Nickel Generic
#-----------------------------------------------------
material[1]:material-name=Ni
material[1]:damping-constant=0.01
material[1]:exchange-matrix[1]=2.757e-21
material[1]:atomic-spin-moment=0.606 !muB
material[1]:uniaxial-anisotropy-constant=5.47e-26
material[1]:material-element=Ni
```

input

```
#-----------------------------------------
# Creation attributes:
#-----------------------------------------
create:crystal-structure=fcc
create:periodic-boundaries-x
create:periodic-boundaries-y
create:periodic-boundaries-z
#-----------------------------------------
# System Dimensions:
#-----------------------------------------
dimensions:unit-cell-size = 3.524 !A
dimensions:system-size-x = 4.0 !nm
dimensions:system-size-y = 4.0 !nm
dimensions:system-size-z = 4.0 !nm
#-----------------------------------------
# Material Files:
#-----------------------------------------
material:file=Ni.mat
#-----------------------------------------
# Simulation attributes:
#-----------------------------------------
sim:temperature=300
sim:minimum-temperature=0
sim:maximum-temperature=800
sim:temperature-increment=25
sim:time-steps-increment=1
sim:equilibration-time-steps=1000
sim:loop-time-steps=1000
```

```
#-----------------------------------------
# Program and integrator details
#-----------------------------------------
sim:program=curie-temperature
sim:integrator=monte-carlo
#-----------------------------------------
# Data output
#-----------------------------------------
output:real-time
output:temperature
output:magnetisation
output:magnetisation-length
output:mean-magnetisation-length
```

Running Vampire

```
rfle500@MacPro:~$ vampire
                                                 _
                                                (_)
          __   _____ _ _ __ ___  _ __  _ _ __ ___
          \ \ / / _` | '_ ` _ \| '_ \| | '__/ _ \
           \ V / (_| | | | | | | |_) | | | |  __/
            \_/ \__,_|_| |_| |_| .__/|_|_|  \___|
                               | |
                               |_|

                  Version 3.0.3 Aug  4 2014 21:00:13

  Licensed under the GNU Public License(v2). See licence file for details.

  Lead Developer: Richard F L Evans <richard.evans@york.ac.uk>

  Contributors: Weijia Fan, Phanwadee Chureemart, Joe Barker,
                Thomas Ostler, Andreas Biternas, Roy W Chantrell

                Compiled with:  GNU C++ Compiler
                Compiler Flags:

================================================================================
Mon Aug  4 21:56:21 2014
================================================================================
Initialising system variables
Creating system
```

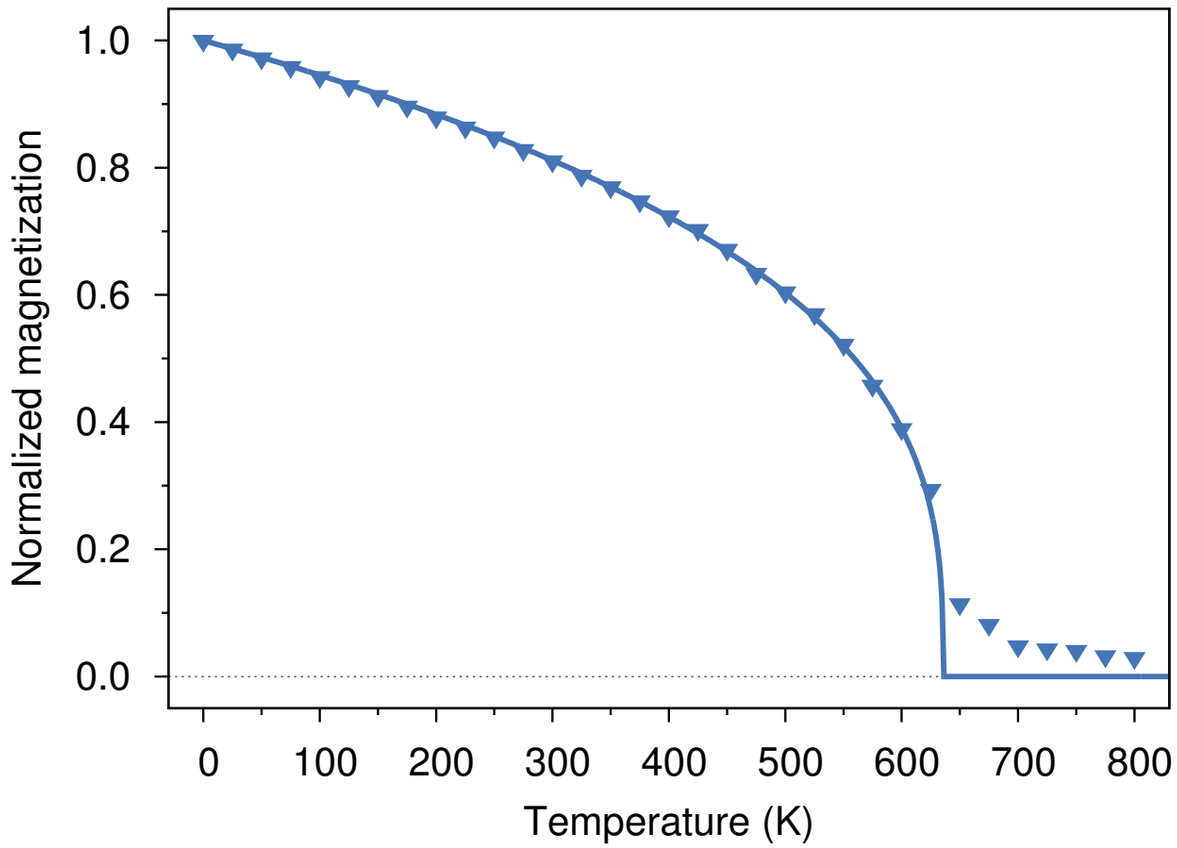
Curie temperature calculation

Calculate phase transition in Ni

Essential temperature dependent property of a magnetic material

$$\mathscr{H} = -\sum_{i<j} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j$$

$$J_{ij} = \frac{3k_B T_c}{\gamma z}$$

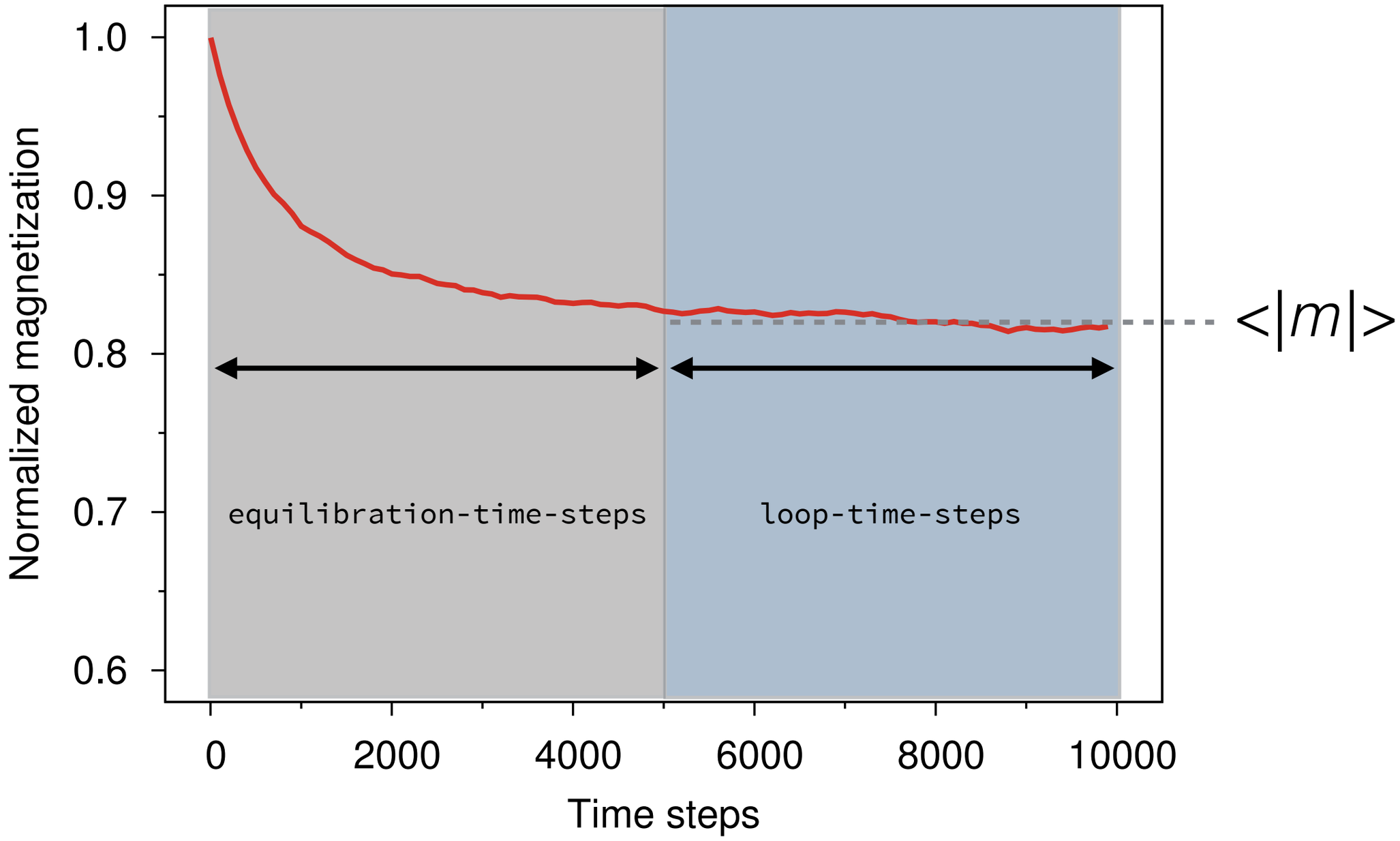


input

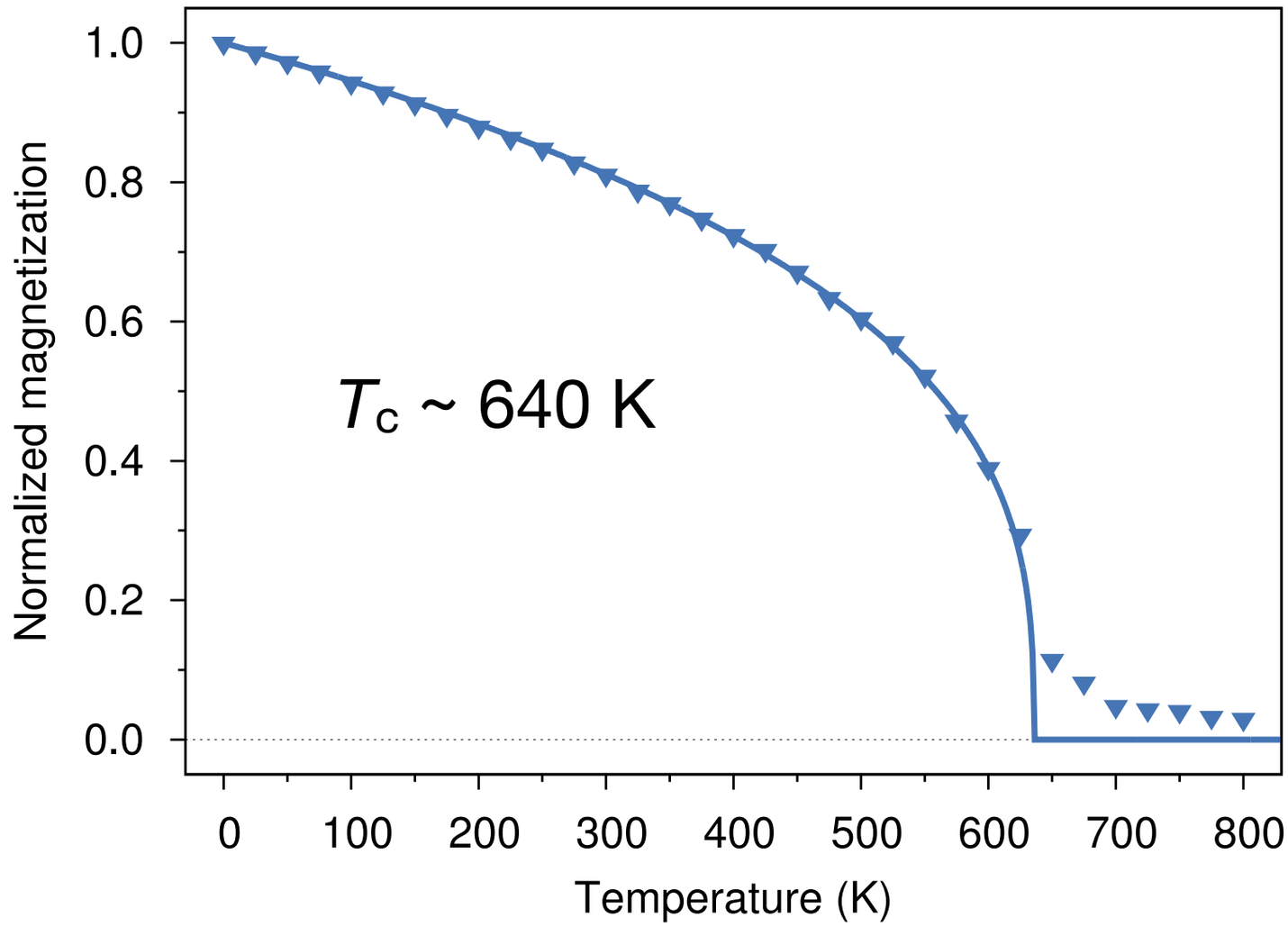
```
#-----------------------------------------
# Creation attributes:
#-----------------------------------------
create:crystal-structure=fcc
create:periodic-boundaries-x
create:periodic-boundaries-y
create:periodic-boundaries-z
#-----------------------------------------
# System Dimensions:
#-----------------------------------------
dimensions:unit-cell-size = 3.524 !A
dimensions:system-size-x = 4.0 !nm
dimensions:system-size-y = 4.0 !nm
dimensions:system-size-z = 4.0 !nm
#-----------------------------------------
# Material Files:
#-----------------------------------------
material:file=Ni.mat
#-----------------------------------------
# Simulation attributes:
#-----------------------------------------
sim:temperature=300
sim:minimum-temperature=0
sim:maximum-temperature=800
sim:temperature-increment=25
sim:time-steps-increment=1
sim:equilibration-time-steps=1000
sim:loop-time-steps=1000
```

```
#-----------------------------------------
# Program and integrator details
#-----------------------------------------
sim:program=curie-temperature
sim:integrator=monte-carlo
#-----------------------------------------
# Data output
#-----------------------------------------
output:real-time
output:temperature
output:magnetisation
output:magnetisation-length
output:mean-magnetisation-length
```

Curie temperature calculation



Curie temperature calculation

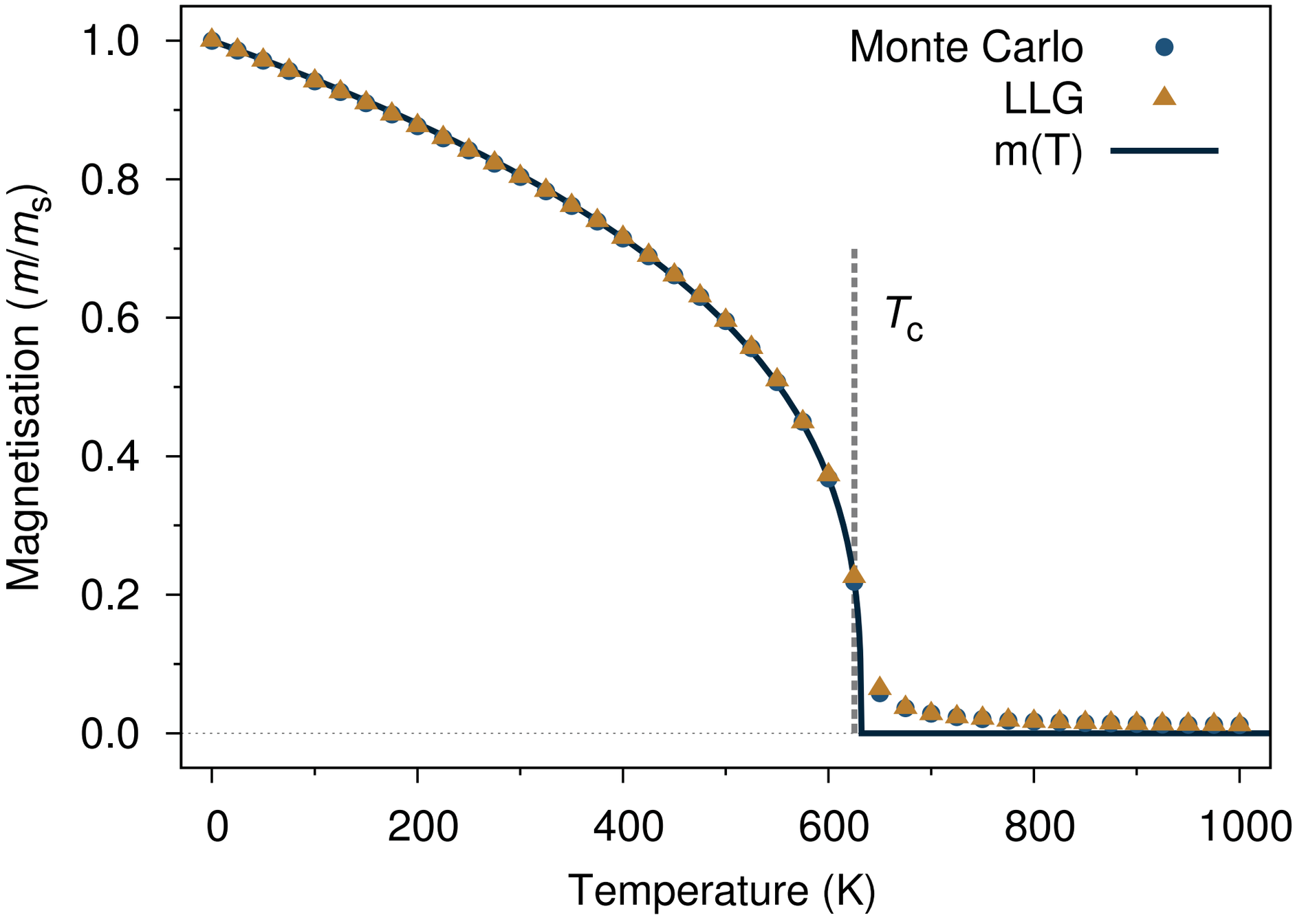


$$m(T) = \left[1 - \left(\frac{T}{T_c}\right)\right]^{\beta}$$

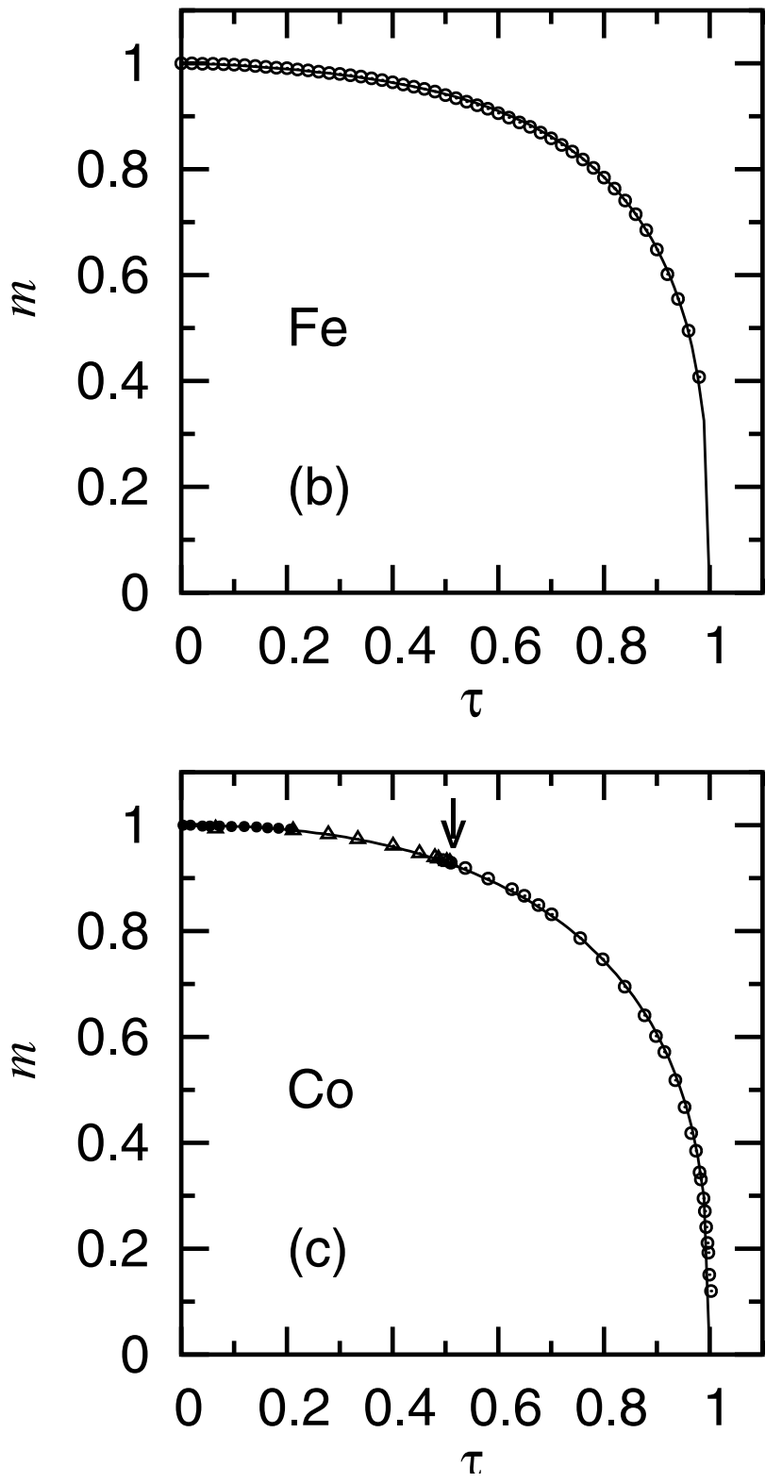
Challenge: reproduce and $M_S(T)$ curve for Nickel

Spin temperature rescaling

Classical spin model *m(T)* simulation



Real ferromagnets: Kuz'min equation



$$m(\tau) = [1 - s\tau^{3/2} - (1 - s)\tau^{p}]^{1/3}$$

Real ferromagnets very
different from classical model
→ problem!

M. D. Kuz'min, Phys. Rev. Lett. **94**, 107204 (2005)

Phenomenological temperature rescaling

Classical model

$$m(T) = (1 - T/T_c)^{\beta}$$

Assume m(T) well fitted by Curie-Bloch equation

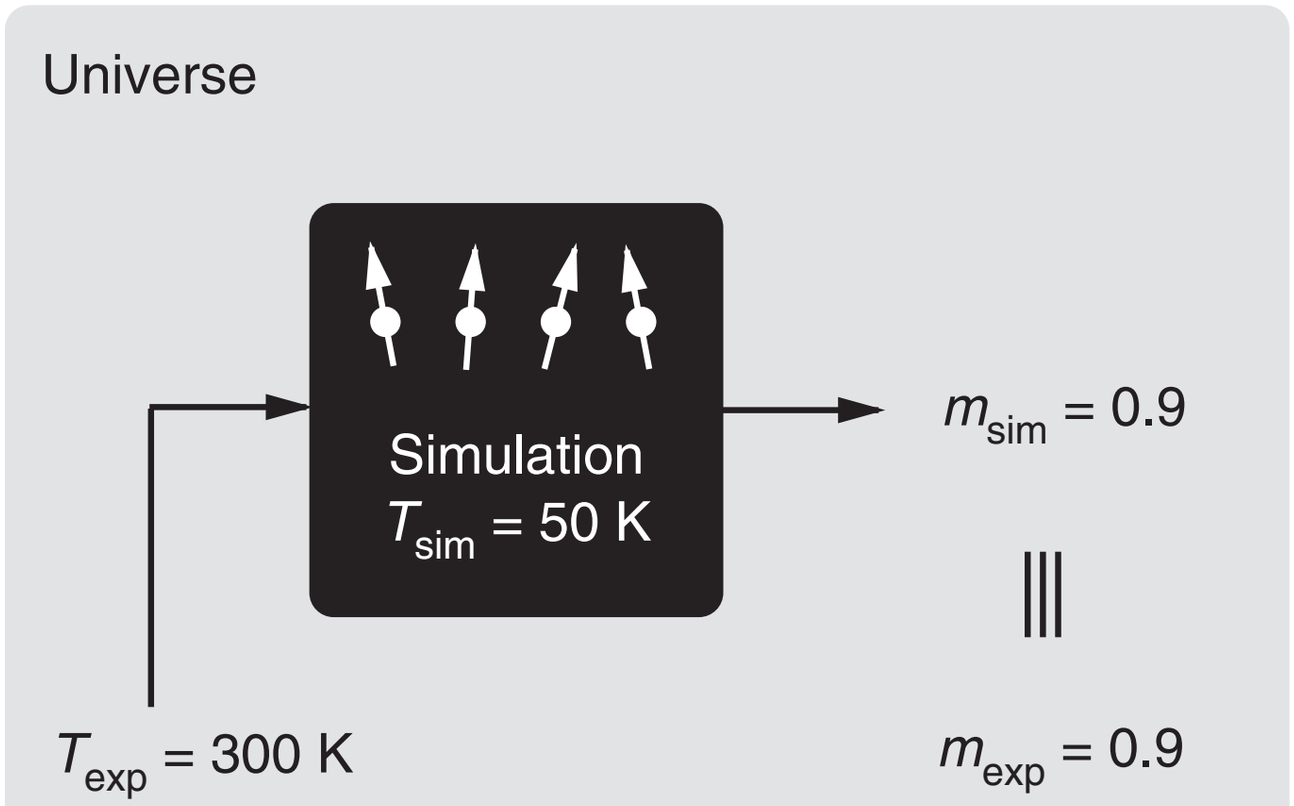
$$m(\tau) = (1 - \tau^{\alpha})^{\beta}$$

Classical model: $\alpha = 1$
Real ferromagnets: $\alpha \neq 1$

Simplest rescaling:

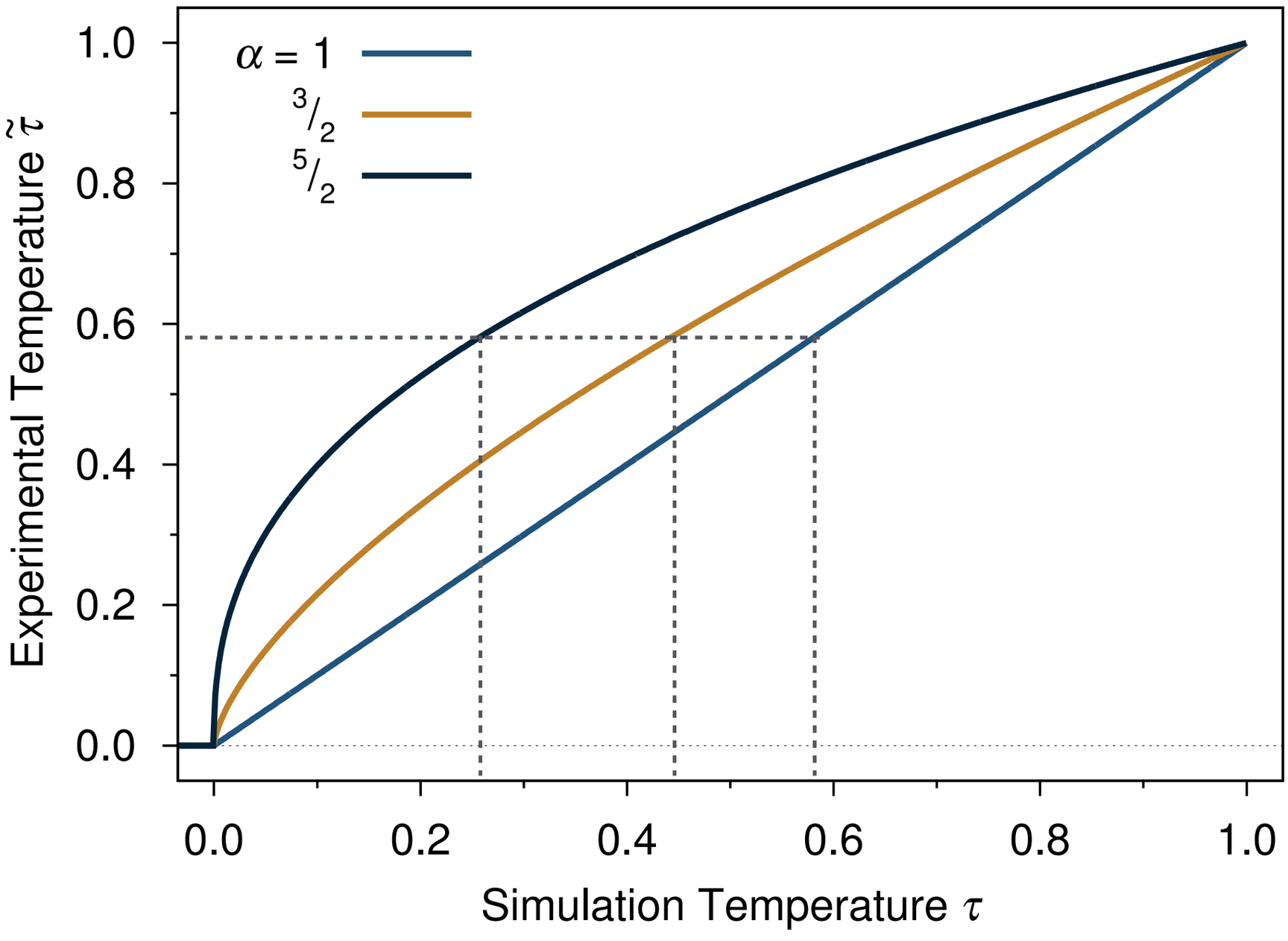
$$\widetilde{\tau} = \tau^{\frac{1}{\alpha}}$$

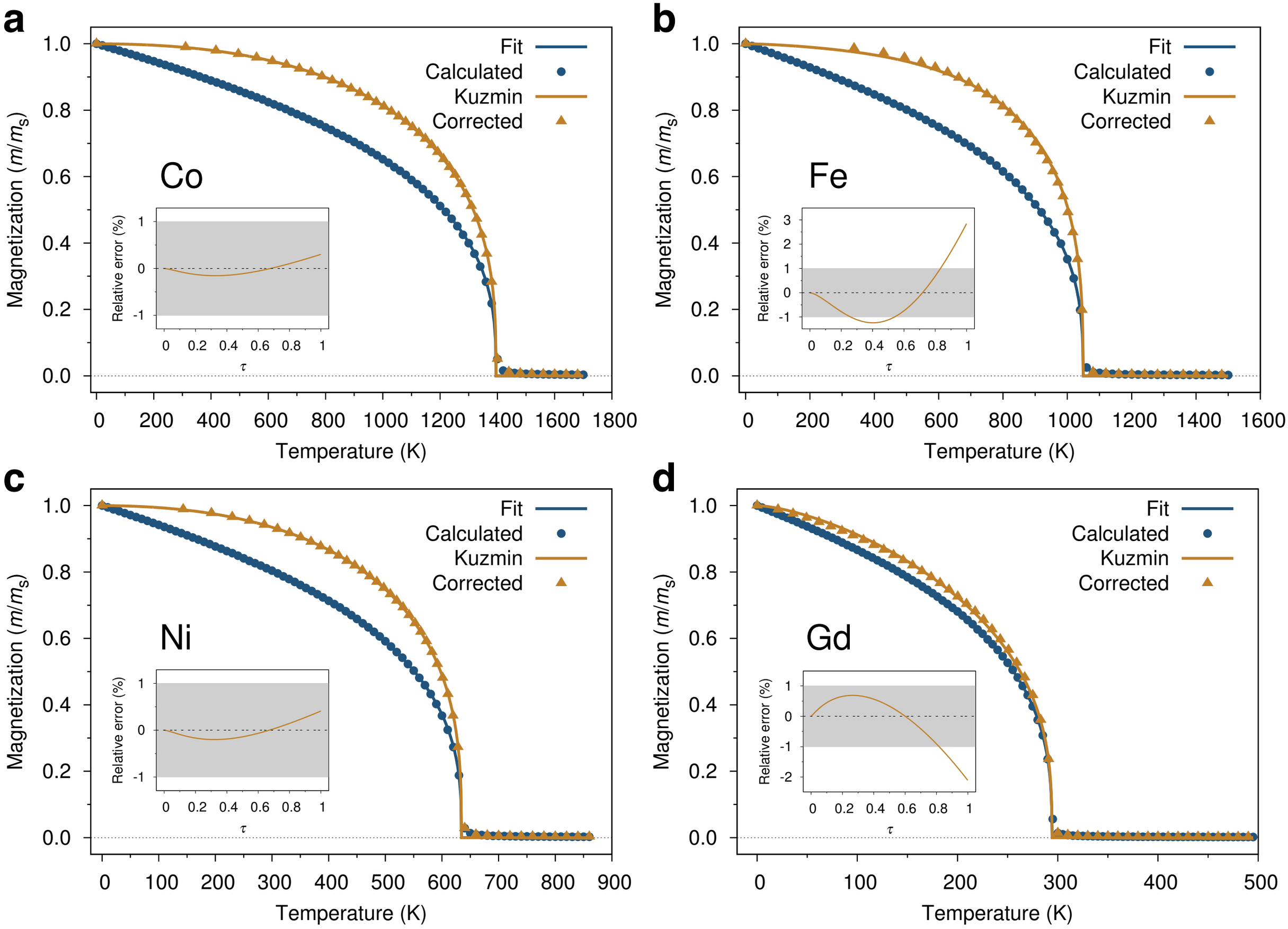
Spin temperature rescaling (STR) method



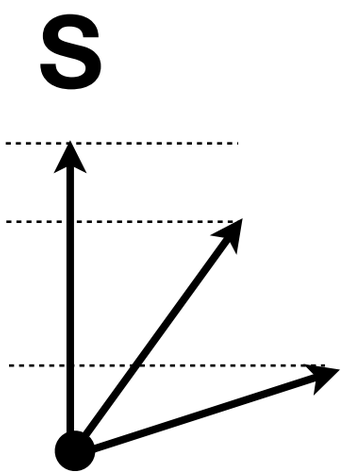
$$\frac{T_{\mathrm{sim}}}{T_{\mathrm{c}}} = \left(\frac{T_{\mathrm{exp}}}{T_{\mathrm{c}}}\right)^{\alpha}$$

Spin temperature rescaling (STR) method

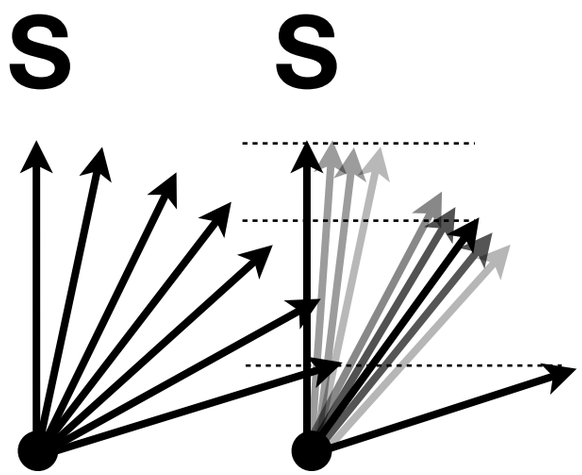



a
b
c
d
Co
Fe
Ni
Gd
Fit
Calculated
Kuzmin
Corrected
Magnetization (m/m_s)
Temperature (K)
Relative error (%)
τ

Physical picture of temperature rescaling



Quantum



Classical Rescaled

Stochastic spin dynamics with rescaled temperature

$$\mathbf{H}_{\mathrm{eff}}^{i} = -\frac{1}{\mu_{\mathrm{s}}}\frac{\partial \mathscr{H}}{\partial \mathbf{S}_{i}} + \mathbf{H}_{\mathrm{th}}^{i}$$

$$\mathbf{H}_{\mathrm{th}}^{i} = \mathbf{\Gamma}(t)\sqrt{\frac{2\lambda k_{\mathrm{B}} T_{\mathrm{sim}}}{\gamma_{e}\mu_{\mathrm{s}}\Delta t}}$$

Applying spin temperature rescaling in VAMPIRE

```
material[1]:temperature-rescaling-exponent=2.322
material[1]:temperature-rescaling-curie-temperature=635.0
```

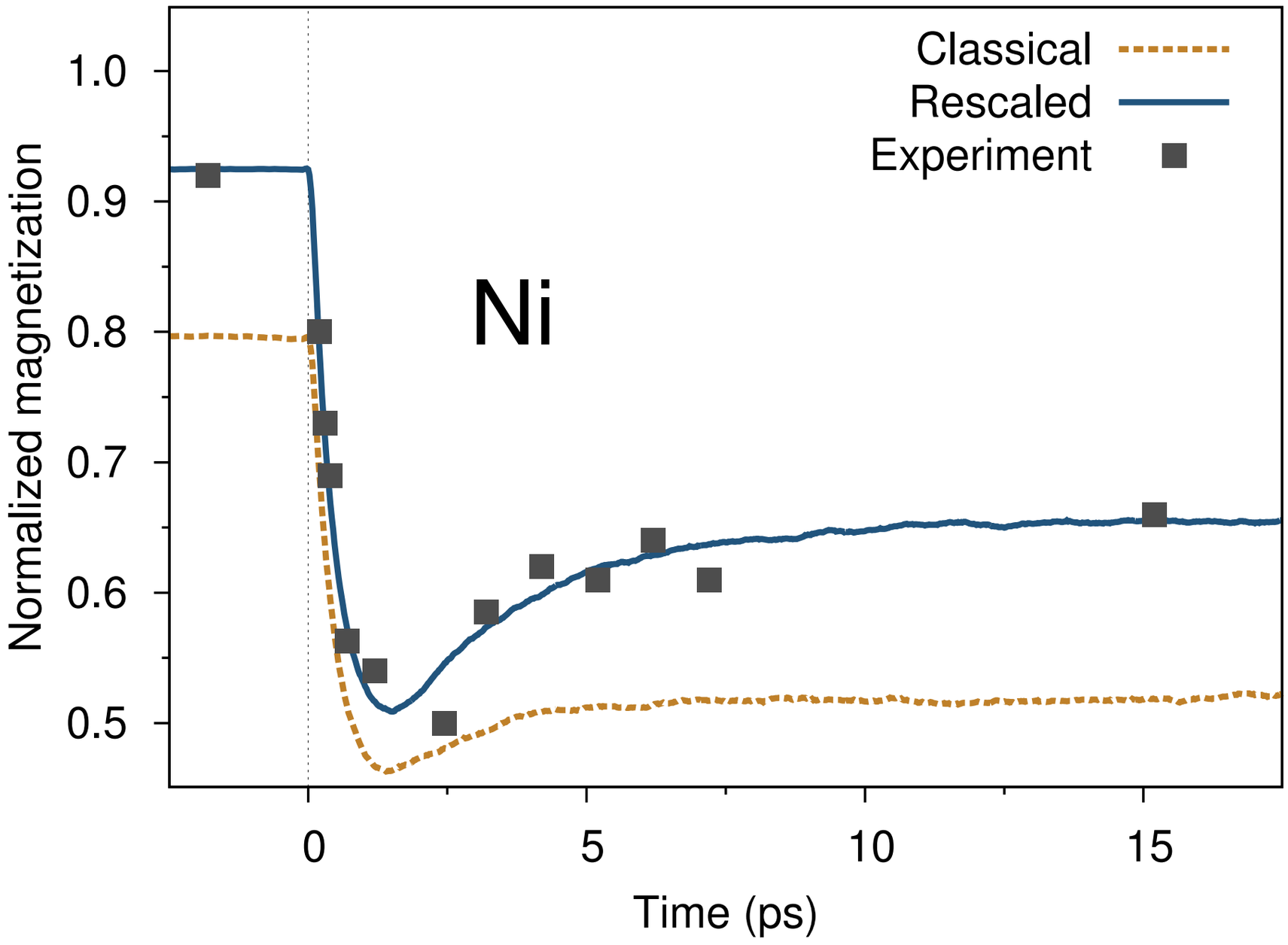
Downsides:

Need to know Cure temperature in advance

Very tricky for more than one sublattice since Tc depends on rescaling -> iterative process

Quantum statistics for a heat bath is the natural solution but has its own complexities…

Ultrafast demagnetization in Ni

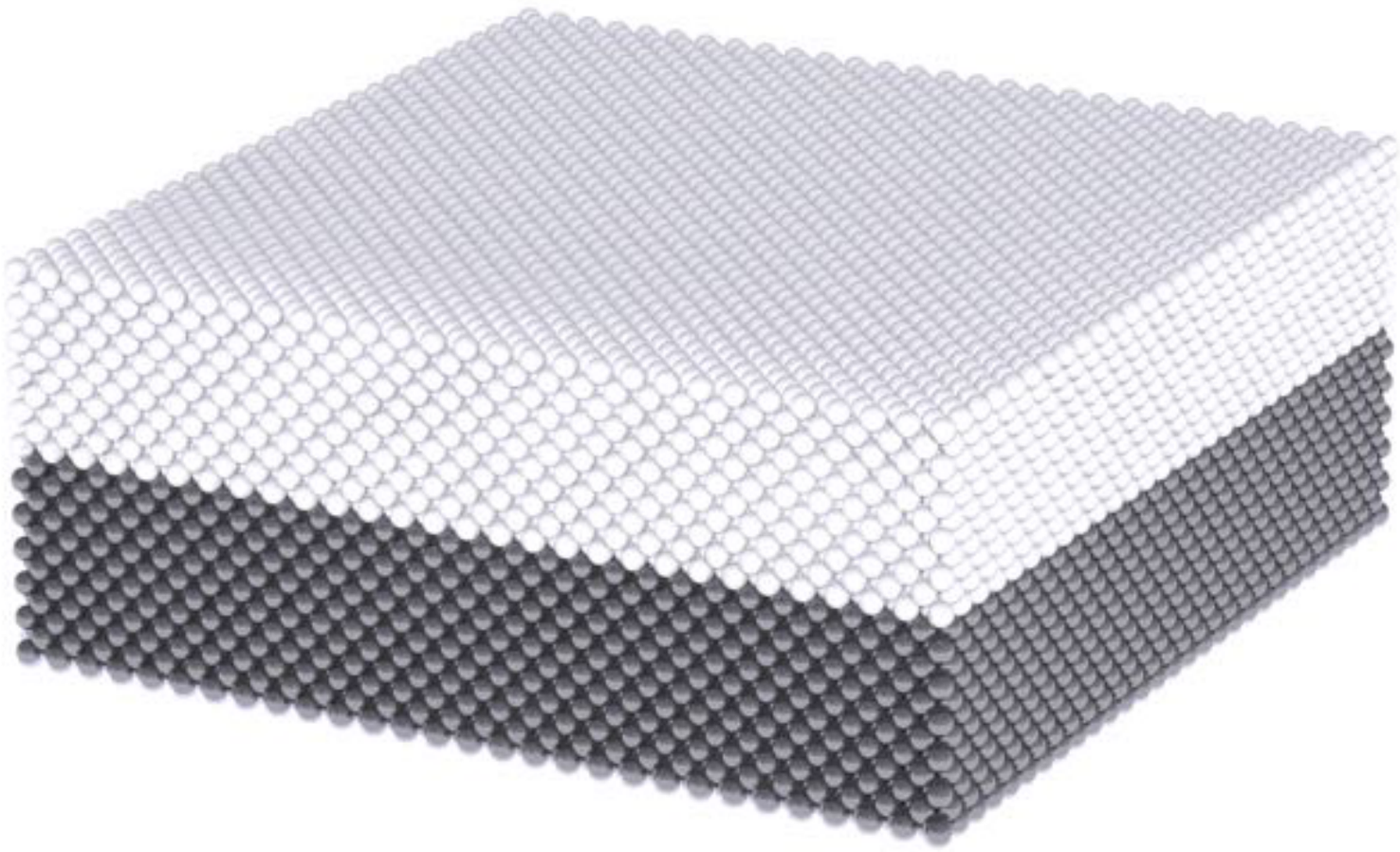


E. Beaurepaire *et al*, Phys. Rev. Lett. **76**, 4250 (1996)

R. F. L. Evans *et al*, Phys. Rev. B **91**, 144425 (2015)

Challenge: reproduce and $M_S(T)$ curve for Nickel including temperature rescaling

Practical 3: System Generation



View your structures with rasmol

Compile vdc (VAMPIRE data convertor) utility

```
make vdc
```

Run the vdc utility

```
/path/to/vdc/vdc
```

View the generated structure with rasmol

```
rasmol -xyz crystal.xyz
```

Enable configuration output

Enable output with default parameters

```
config:atoms
```

Change output rate (as multiple of main data output rate)

```
config:atoms-output-rate = 100
```

Select slices of the data

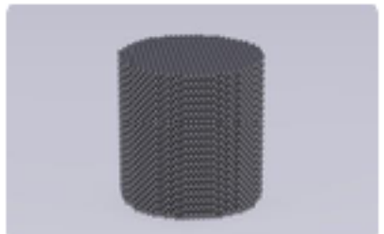
```
config-atoms
```

http://vampire.york.ac.uk/tutorials/

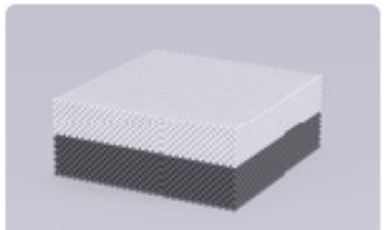
Tutorials

Vampire is a large software package with a lot of options, and so the following tutorials give a brief introduction to using Vampire. As time allows more tutorials will be added here.

Introductory

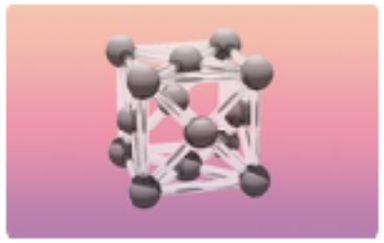


System generation

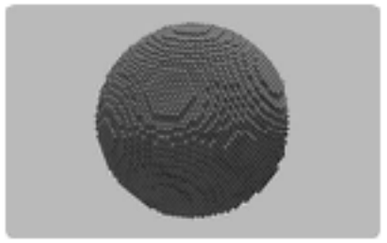


Bilayer

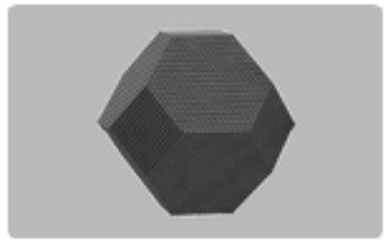
System generation



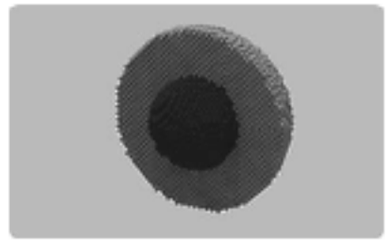
Crystal structures



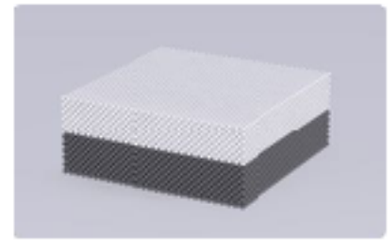
Sphere



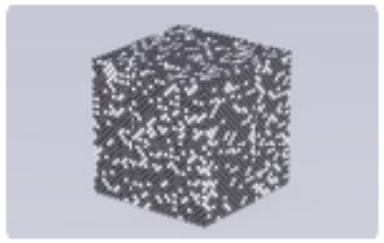
Truncated octahedron



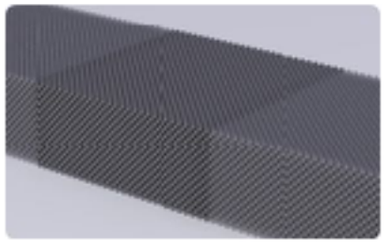
Core-shell



Multilayers



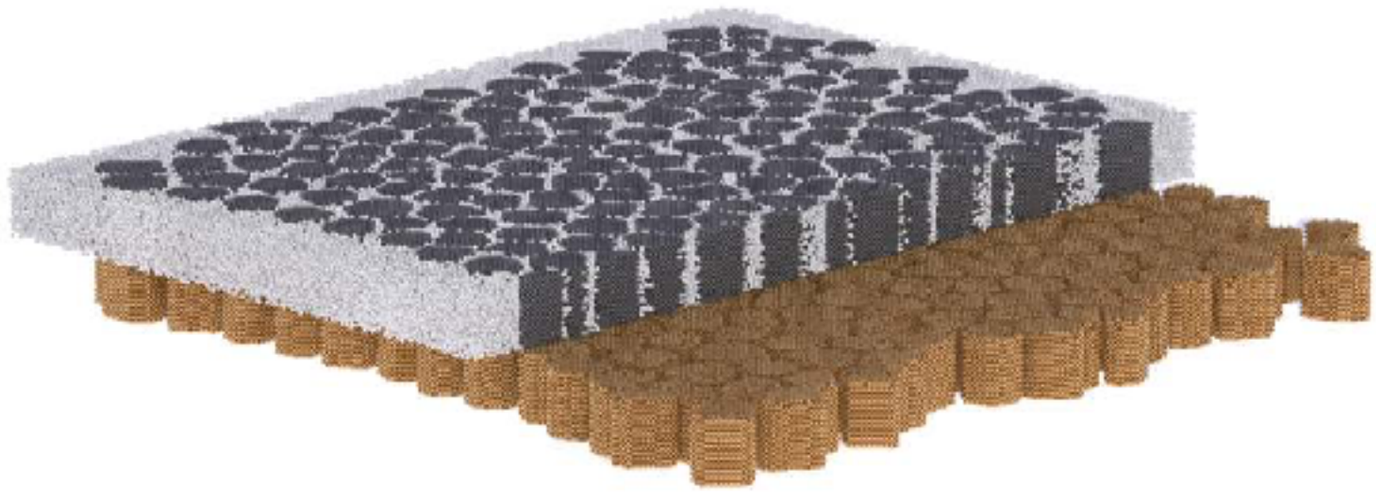
Random alloys



Periodic boundaries

Challenge: generate a single material structure and core-shell structure

Challenge: generate a voronoi grain structure



```
create:voronoi-film
create:voronoi-rounded-grains

dimensions:particle-size = 6.0 !nm
dimensions:particle-spacing = 1.0 !nm
```

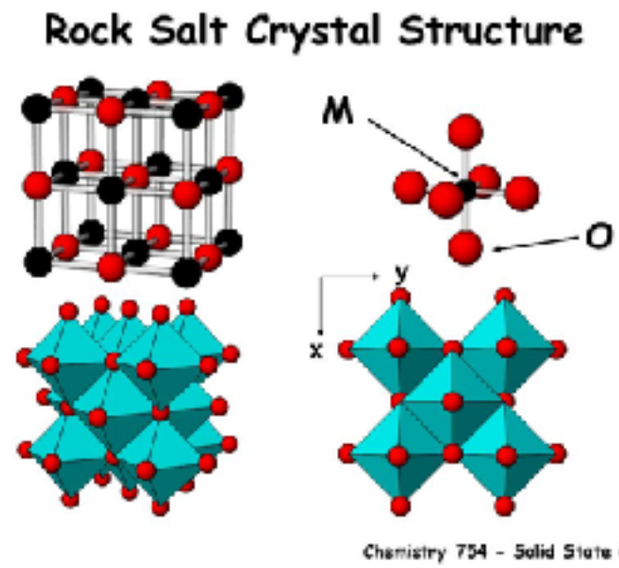
(Also compatible with multilayers/core shell structures)

Unit cells with more than one material

Develop branch has new feature better supporting unit cells (and unit cell files) with more than one material

Example Rocksalt (CsCl) structure now built-in

```
create:crystal-structure = rocksalt
```



Associate different materials in unit cell with different materials in VAMPIRE

```
material[1]:unit-cell-category = 1
material[2]:unit-cell-category = 2
```

Non-magnetic materials

Develop branch now has support for non-magnetic atoms for improved visualisation and also effects such as SO coupling

Can identify a material as non-magnetic (removed from simulation)

```
material[1]:non-magnetic = remove
```

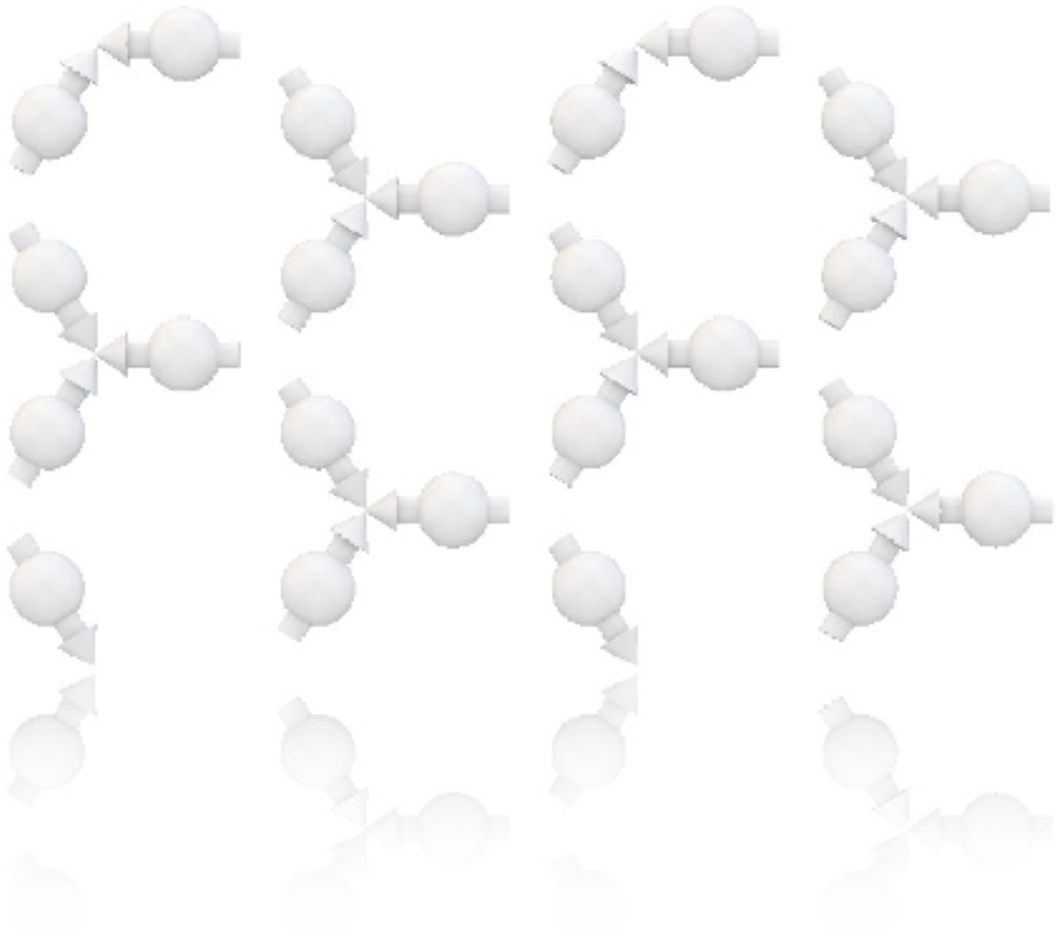
Or non-magnetic (keep in simulation but not in statistics/dipole)

```
material[1]:non-magnetic = keep
```

Using vdc utility removed atoms are retained for visualisation

Challenge: generate a core-shell nanoparticle with a multiple material unit cell

Practical 4: Statistics



Magnetization statistics

VAMPIRE contains a range of methods for calculating statistical properties

Can output total magnetization specified in input file

```
output:magnetisation
```

Data output is in 4 columns (unit vector and length)

$$\hat{m}_x \; \hat{m}_y \; \hat{m}_z \; |m|$$

Can control rate of output as a multiple of increment

```
output:output-rate = 10
```

Average statistics

Average (mean) data can also be selected (seen earlier)

```
output:mean-magnetisation-length
```

Generally want a time step increment of 1-5 (rate at which statistics are calculated)

Time series calculations produce a running average (useful for convergence)

Loop calculations produce average at end of loop increment (temperature, field etc)

Material/height statistics

Can slice magnetization data by material and/or height

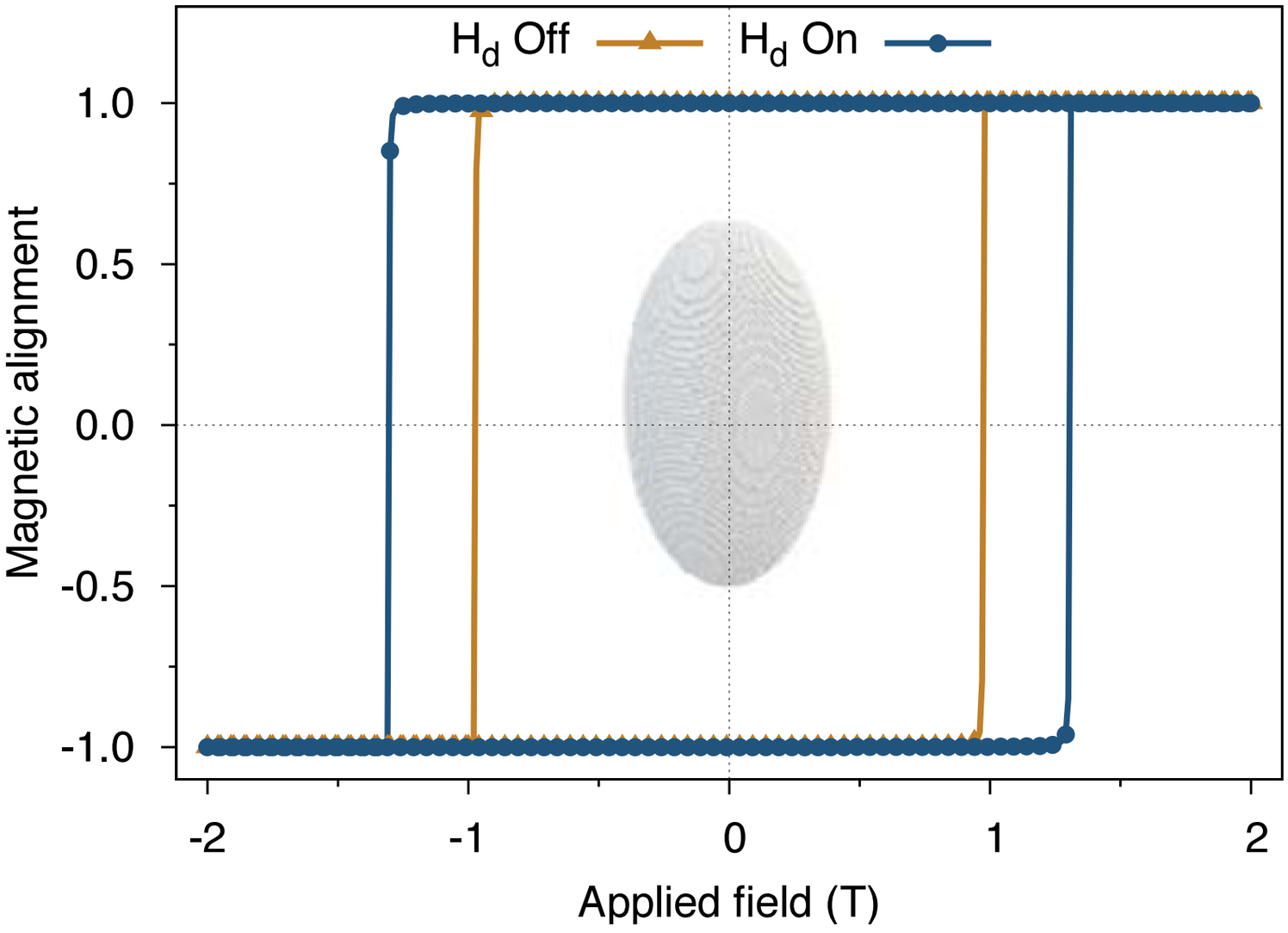
```
output:magnetisation
output:material-magnetisation
output:height-magnetisation
output:height-material-magnetisation
```

Very useful for analysing sublattice magnetization in the case of multiple materials or domain wall processes

Challenge:

Calculate sublattice M(T) curves for rock salt structure with two materials and different exchange constants

Practical 5: Hysteresis simulations



Hysteresis calculations

Generally a ‘slow’ process - typically 10s of nanoseconds

For comparison with experiment, use high damping limit, $\lambda = 1$

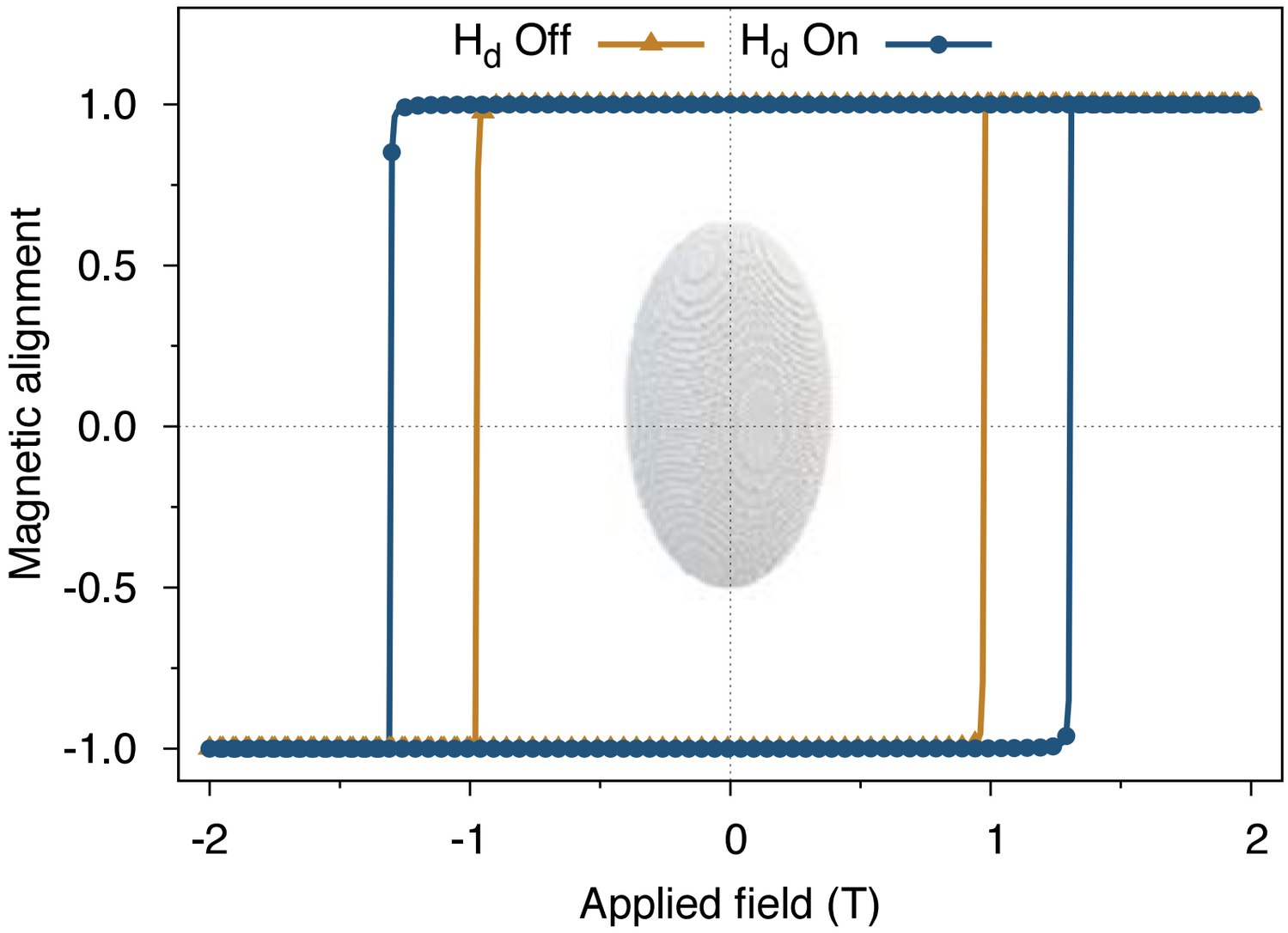
Coercivity **strongly** field rate dependent - slower is better!

input file

```
sim:loop-time-steps=100000
sim:program=hysteresis-loop
sim:integrator=llg-heun
sim:time-step=1.0e-15
sim:temperature = 0
sim:equilibration-applied-field-strength = 2.0 !T
sim:maximum-applied-field-strength = 2.0 !T
sim:applied-field-strength-increment = 0.01 !T
sim:applied-field-angle-phi = 0.1 # (degrees from z)

output:real-time
output:applied-field-strength
output:applied-field-alignment
output:magnetisation
```

hysteresis-loop program



Cycles field from $+H_{max}$ to $-H_{max}$ in a user defined increment

Calculates dynamic response of the magnetisation to the field

Challenge:

Generate M(H) curves for a (1 nm)^3 sample at 0K for different field rates, anisotropy and damping

Questions

What happens to the hysteresis loop if you change loop-time?

How many steps (field rate) do you need to reach the limit $H_c = 2k_u/\mu_s$

What effect does changing the field angle have?

What effect does the damping have?