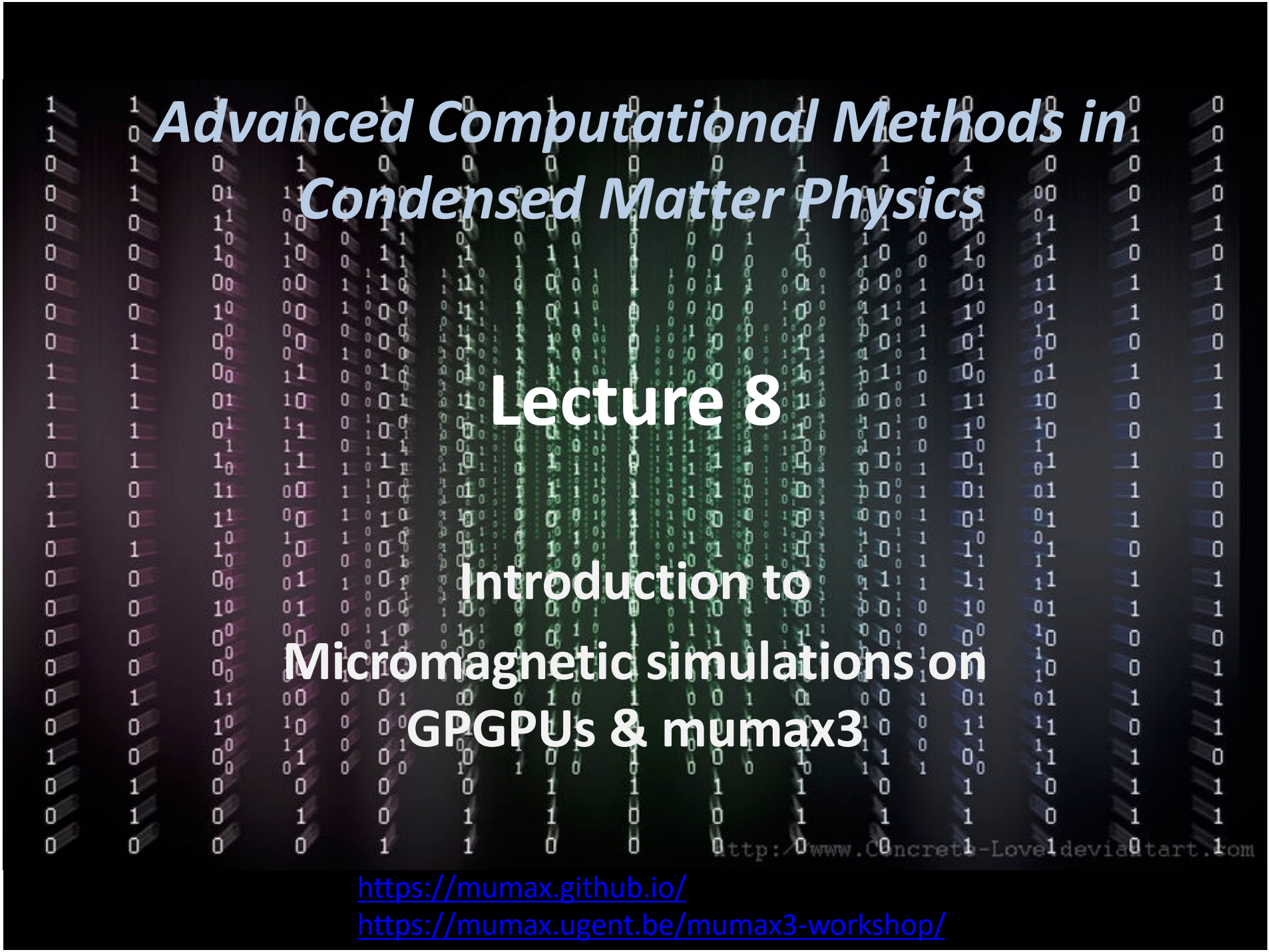


Advanced Computational Methods in Condensed Matter Physics

Lecture 8

Introduction to Micromagnetic simulations on GPGPUs & mumax3

https://mumax.github.io/
https://mumax.ugent.be/mumax3-workshop/

Motivation

Micromagnetics

- prediction of magnetic behaviors at sub-micrometer length scales (1nm – 1µm)
- continuum approximation → atomic structure of the material can be ignored
- but small magnetic structures like domain walls or vortices are resolved

Micromagnetic calculations can be

- static, by minimizing the magnetic energy,
- dynamic , by solving the time-dependent dynamical equation, time-scale: ps

Origins

- work by Lev Landau and Evgeny Lifshitz on antidomain walls (1935)
- expanded by William Fuller Brown Jr., who termed "*micromagnetics*" in 1958
- computational methods developed after 1970 for magnetic recording media for data

Static Micromagnetics

Magnetic energy minimization:

$$E = E_{exch} + E_{anis} + E_Z + E_{demag} + E_{DMI} + E_{m-e}$$

With constraint $|\boldsymbol{m}| = 1$ with $\boldsymbol{m} = \boldsymbol{M}/M_s$ (M_s saturation magnetization)

Contributions:

- E_{exch}: Exchange energy - phenomenological continuum description of quantum-mechanical exchange interaction
- E_{anis}: Anisotropy energy - interplay of crystal structure and spin-orbit interaction
- E_Z: Zeeman energy - interaction energy between magnetization and externally applied field
- E_{demag}: Demagnetization energy - demagnetizing field is magnetic field of magnetic sample itself
- E_{DMI}: Dzyaloshinskii–Moriya Interaction energy due to breaking of crystalline inversion symmetry → magnetization perpendicular to its neighbors
- E_{m-e}: Magnetoelastic Energy energy storage due to elastic lattice distortions

Yields, e.g., stable magnetic states, hysteresis curves, phase diagrams, domain wall profiles

Dynamic Micromagnetics

Goal: predict time evolution of the magnetic configuration, like spin waves, domain wall motion, spin transfer torques, vortex excitation

Important in non-steady conditions like AC or pulsed fields

→ Solve the **Landau–Lifshitz–Gilbert equation** (LLG), a PDE for the temporal evolution of the magnetization under an effective field, the local field felt by the magnetization

$$\boldsymbol{H}_{eff} = -\frac{1}{\mu_0 M_s}\frac{d^2E}{d\boldsymbol{m}dV}$$

with variational magnetic energy

$$dE = -\mu_0 M_s \int_V (d\boldsymbol{m}) \cdot \boldsymbol{H}_{eff} dV$$

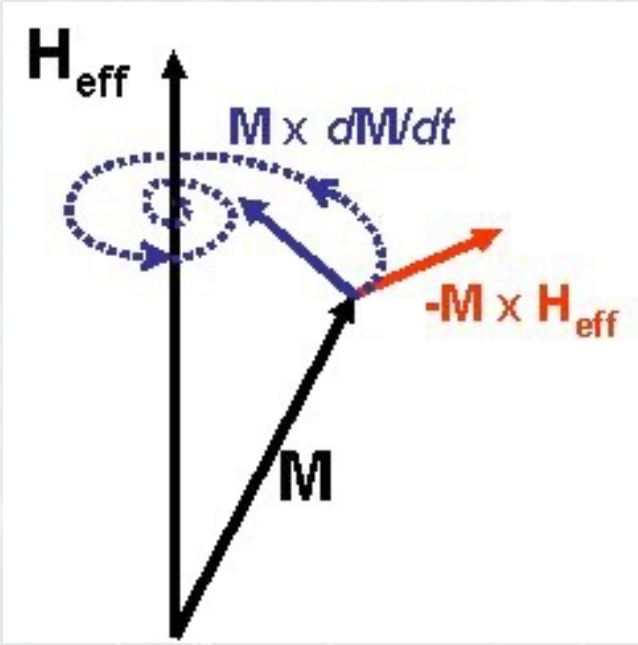
This gives (w/o DMI and e-m terms):

$$\boldsymbol{H}_{eff} = \frac{2A}{\mu_0 M_s}\nabla^2\boldsymbol{m} - \frac{1}{\mu_0 M_s}\frac{\partial F_{anis}}{\partial \boldsymbol{m}} + \boldsymbol{H}_a + \boldsymbol{H}_d$$

LLG equation

Magnetization dynamics is given by:

$$\dot{\boldsymbol{m}} = -\frac{\gamma}{1+\alpha^2}\left[\underbrace{\boldsymbol{m}\times\boldsymbol{H}_{eff}}_{\text{precession}} + \underbrace{\alpha\,\boldsymbol{m}\times\left(\boldsymbol{m}\times\boldsymbol{H}_{eff}\right)}_{\text{damping}}\right]$$



Additional terms:

- Spin transfer torque (magnetic layer influenced by spin-polarized currents → e.g. magnetic memory)

$$\dot{\boldsymbol{m}} = -\frac{\gamma}{1+\alpha^2}\left[\boldsymbol{m}\times\boldsymbol{H}_{eff} + \alpha\,\boldsymbol{m}\times\left(\boldsymbol{m}\times\boldsymbol{H}_{eff}\right)\right] + \tau_{STT}$$

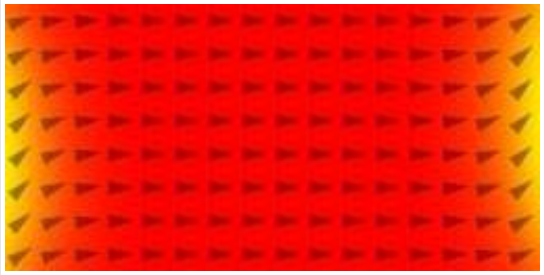
- Random effective field (thermal noise) $\boldsymbol{H}_{eff} \rightarrow \boldsymbol{H}_{eff} + \boldsymbol{H}_{th}$

$$\langle \boldsymbol{H}_{th}(\boldsymbol{r},t)\rangle = 0$$

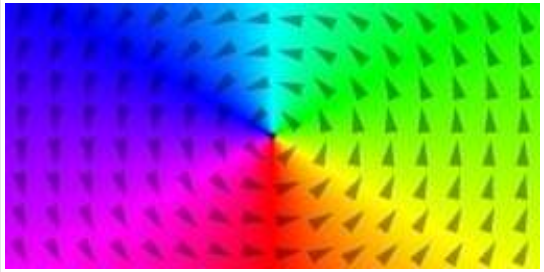
$$\langle \boldsymbol{H}_{th}(\boldsymbol{r},t), \boldsymbol{H}_{th}(\boldsymbol{r}',t')\rangle = \frac{2k_B T\alpha}{M_s\gamma}\delta(\boldsymbol{r}-\boldsymbol{r}')\delta(t-t')$$

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

Examples for statics



Quasi-uniform state



Vortex

Néel Skyrmion

Numerical energy minimization by

1. Standard minimization scheme (*e.g. steepest gradient*)
2. LLG equation with strong damping *(=removing the precession term)*

$$\dot{\boldsymbol{m}} = -\boldsymbol{m} \times \left(\boldsymbol{m} \times \boldsymbol{H}_{eff}\right)$$

Dynamics → mumax^3 for LLG

What is mumax^3?

Free finite-difference based micromagnetic simulation package

- GPU-accelerated nvidia GPU required
- Latest official release mumax3.11 (Aug 13, 2020)
- Documented API: mumax.github.io
- Open source (GPLv3) github.com/mumax/3
- Mainly written in Go
- CUDA C kernels for heavy lifting
- Scripting language + Web GUI
- Well tested (unit tests + NIST standard problems)

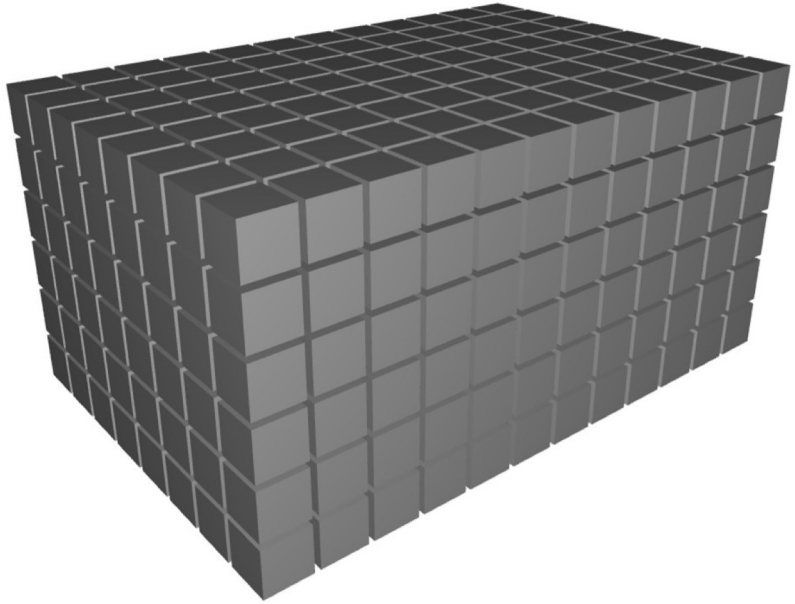
How to use?

Uses a scripting language which is a subset of *golang*

```
// saturation magnetization
Msat = 5e6

// declare new variable
Freq := 1e9

for i:=0; i<10; i++ {
        print(i)
}

if 1+8 == 9 {
        print("Of course 1+8=9")
}
```

Source: mumax3 tutorial

Discretization

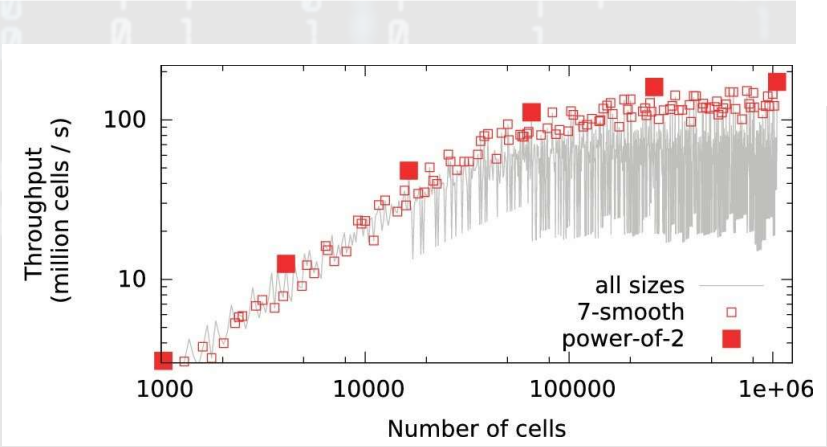


- Rectangular simulation box (origin in the center)
- Single regular rectangular grid
- Uniform magnetization inside cell $\boldsymbol{m}_{ijk} = \boldsymbol{m}(x_i, y_j, z_k)$
- Cell size < exchange length

```
setgridsize(256,64,1)
setcellsize(1e-9,1e-9,1e-9)
```

- PBC values are number of virtual repetitions of the simulation box used to calculate dipolar interactions

```
setpbc(4,0,0)
```



The cuda fft library (used for the computation of the demag field) is highly optimized for grid size dimensions with small prime factors (less than 7).

For the discretization in mumax, one should choose the cell size such that:

- It is larger than the atomistic scale (>=1nm)
- Smaller than the exchange length is $l_{ex} = (A/K_m)^{1/2}$, where A is the exchange stiffness constant and K_m is a magnetostatic energy density: $K_m = {}^1/_2\mu_0 M_s^2$ (SI) or $2\pi M_s^2$ (cgs emu)
- The maximum angle of the magnetization between neighboring cells (maxangle) is less than 20° (most of the time)

In general, the micromagnetic continuum description works on scales from 1nm to 1μm

Shapes

- A shape can be considered as a function $f: \mathbb{R}^3 \rightarrow \{true, false\}$ where

$$f(x,y,z) = \begin{cases} true \text{ if } (x,y,z) \text{ in shape} \\ false \text{ otherwise} \end{cases}$$

- Most shapes do not depend on the grid
- Default location: center of universe (0,0,0)
- Large set of predefined basic shapes
- Define new shapes by combining and modifying the basic shapes
- Shapes are useful for different tasks:
  - The geometry
  - Defining regions
  - To set locally an initial magnetization

Shape functions and example

Shapes

Cell(j,k,l)
Circle(diameter)
Cone(diameter,height)
Cuboid(Lx,Ly,Lz)
Cylinder(diameter,height)
Ellipse(a,b)
Ellipsoid(a,b,c)
ImageShape(filename)
Layer(i)
Layers(i1,i2)
Rect(Lx,Ly)
Square(L)
Xrange(xmin,xmax)
Yrange(ymin,ymax)
Zrange(zmin,zmax)

Shape methods

Transl(dx,dy,dz)
Scale(sx,sy,sz)
RotX(angle)
RotY(angle)
RotZ(angle)
Repeat(dx,dy,dz)

Add(shape)
Sub(shape)
Inverse()
Intersect(shape)
Xor(shape)

```
// Rotated cheese example

d  := 200e-9
sq := square(d)

h     := 50e-9
hole  := cylinder(h, h)
hole1 := hole.transl(100e-9, 0, 0)
hole2 := hole.transl(0, -50e-9, 0)

cheese := sq.sub(hole1).sub(hole2)
cheese = cheese.rotz(pi/6)
```

rotated cheese

Geometry

Optionally a magnet Shape other than the full simulation box can be specified

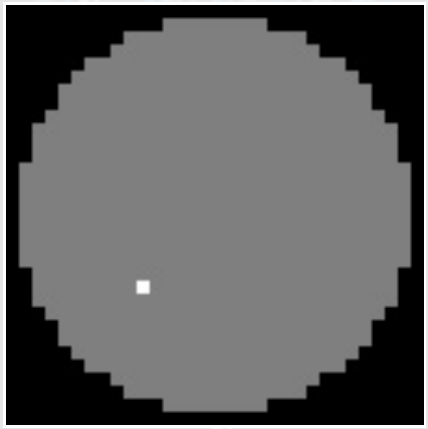
```
// Ring geometry example

setGridsize(100, 100, 10)
setCellsize(1e-9,1e-9,1e-9)

ring := circle(100e-9).sub(circle(50e-9))

setgeom(ring)

save(geom)
```

Regions

256 regions in total (index 0→ 255)

- Each cell is assigned to a single region (default region id is 0)
- Each region has its own set of material parameters
- Two ways to set the region id in cells:
  1. Set region id of a single cell
  2. Set region id of all cells in a shape

```
SetGridSize(32,32,1)
SetCellSize(1,1,1)

// Set region id of cells in a circle to 1
DefRegion(1, circle(30))

// Set region id of cell (10,10,0) to 2
DefRegionCell(2, 10, 10, 0)

Save(regions)
```



Material Parameters

Material parameters are assigned to the 256 regions

- Material parameters can be functions of time
- There are vector and scalar material parameters
- Material parameters are predefined, they can not be created

```
// Assigning to a material parameter sets a value in all regions:
Msat  = 800e3
AnisU = vector(1, 0, 0)

// When regions are defined, they can also be set region-wise:
Msat.SetRegion(0, 800e3)
Msat.SetRegion(1, 540e3)

// Material parameters can be functions of time as well:
f := 500e6
Ku1 = 500 * sin(2*pi*f*t)
```

Excitations

- An excitation is a regional material parameter
- Additionally, one can add an arbitrary number of time- and space-dependent vector fields of the form: $g(x,y,z)*f(t)$

```
B_ext = vector(0,0,1)

B_ext.Add(LoadFile("antenna.ovf"), sin(2*pi*f*t))

B_ext.removeExtraTerms()
```

Initial magnetization

Different ways to set the magnetization:

Here, a ‘config’ is an object which represents a magnetization configuration

```
m = config
m.LoadFile(filename)
m.SetRegion(regionId, config)
m.SetInShape(shape, config)
m.SetCell(j,k,l, vector)
```

Config

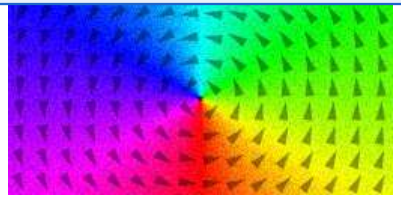
Uniform(mx, my, mz)
RandomMag()
RandomMagSeed(seed)
TwoDomain(
  mx1, my1, mz1,
  my2, my2, mz2,
  mx3, my3, mz3)
Vortex(circ, pol)
AntiVortex(circ, pol)
VortexWall(mxLeft, mxRight, circ, pol)
NeelSkyrmion(charge, pol)
BlochSkyrmion(charge, pol)
Conical(kVec, coneDir, coneAngle)
Helical(kVec)

Config methods

Transl(dx,dy,dz)
Scale(sx,sy,sz)
Add(ratio, config)
RotZ(angle)

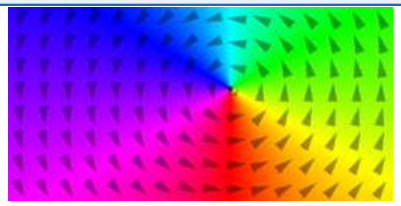
```
m = uniform(1,1,0)
```

```
m=Vortex(1,-1).Add(0.1,randomMag())
```



```
M = BlochSkyrmion(1, 1)
```

```
m=Vortex(1,-1).transl(100e-9,50e-9,0)
```



```
m = uniform(1, 1, 1)
m.setInShape(cylinder(400e-9,100e-9), vortex(1,-1))
```

Output

3 output media:

- log file for input, logging and printing
- table.txt (t, mx, my, mz, …)

```
tableadd(E_total)
tableaddvar(myVar,”myVar”,”unit”)
tablesave() // write single line
tableautosave(1e-12) // write periodically
```

- .ovf files for scalar and vector fields

```
save(Edens_total)
saveas(Edens_total,”edens.ovf”)
autosave(Edens_total, 1e-10) // write perdiodically
```

OVF: Object Vector Field, can be converted to other formats using mumax3-convert, e.g., for paraview compatible format:

mumax3-convert -vtk binary m*.ovf

Run/Relax/Minimize

- Solving the LLG equation (time integration)

```
run(timeperiod)
steps(100)
runWhile(condition)
```

- Minimizing the energy

```
relax()    // LLG without precession
Minimize() // steepest descent[1]
```

```
// WARNING: ADVANCED SETTINGS
// In most cases, these settings can be ignored

// Set the solver:
//   1:Euler, 2:Heun, 3:Bogaki-Shampine, 4: Runge-Kutta(RK45),
//   5:Dormand-Prince(the default), 6:Fehlberg, -1:Backward Euler
SetSolver(5)

// set timestep
fixdt = 0 // if 0 (default): use adaptive timestep

// Advanced settings for adaptive timestep (default values are given)
Headroom = 0.8 // headroom dt correction
MaxDt = 0      // if 0, no maximal timestep
MinDt = 0      // if 0, no minimal timestep
MaxErr = 1e-5  // maximum allowed error/step

// Advanced settings for minimizer (default values are given)
MinimizerSamples = 10 //Number of max dM for convergence check
MinimizerStop = 1e-6  //Stopping max dM for Minimize
```

Interaction terms

Effective field terms

- Demagnetization
- Exchange
- Anisotropy
- Dzyaloshinskii-Moriya
- External field
- Thermal field
- Custom field

Spin transfer torques

- Zhang-Li STT
- Slonczewski STT

Demagnetization/Saturation

Demagnetization energy density

$$\varepsilon = -\frac{\mu_0}{2} M_s \boldsymbol{m} \cdot \boldsymbol{H}_{demag}$$

Regional Material Parameters	
`Msat`	Saturation magnetization (A/m)
`NoDemagSpins`	Disable magnetostatic interaction per region (default=0, set to 1 to disable).

Output Quantities	
`B_demag`	Magnetostatic field (T)
`Edens_demag`	Magnetostatic energy density (J/m3)
`E_demag`	Magnetostatic energy (J)

Other functionalities	
`EnableDemag`	Enables/disables demag (default=true)
`SetPBC`	Sets the number of repetitions in X,Y,Z to create periodic boundary conditions. The number of repetitions determines the cutoff range for the demagnetization.
`DemagAccuracy`	Controls accuracy of demag kernel

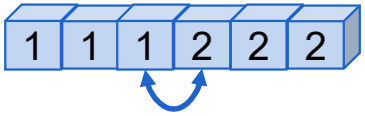
Exchange field

Exchange energy density

$$\varepsilon = \mathrm{A}\,(\nabla \boldsymbol{m})^2$$

Harmonic mean for inter-region exchange coupling (default behavior)

$$\frac{A}{M_s} = 2\,\frac{\frac{A_1}{M_{s_1}}\frac{A_2}{M_{s_2}}}{\frac{A_1}{M_{s_1}} + \frac{A_2}{M_{s_2}}}$$



Regional Material Parameters	
`Aex`	Exchange stiffness (J/m)
`Msat`	Saturation magnetization (A/m)

Output Quantities	
`B_exch`	Exchange field (T)
`Edens_exch`	Total exchange energy density (including DMI) (J/m3)
`E_exch`	Total exchange energy (including DMI) (J)
`MaxAngle`	Maximum angle between exchanged coupled spins (rad)

Other functionalities	
`Ext_InterExchange`	Sets exchange coupling between two regions
`Ext_ScaleExchange`	Re-scales exchange coupling between two regions

Anisotropy field

Uniaxial anisotropy energy density

$$\varepsilon = -K_1 (\hat{u} \cdot \boldsymbol{m})^2 - K_2 (\hat{u} \cdot \boldsymbol{m})^4$$

Similar expression for cubic anisotropy energy density [1]

Regional Material Parameters	
`anisU`	Uniaxial anisotropy direction
`Ku1, Ku2`	Uniaxial anisotropy constants (J/m3)
`AnisC1, AnisC2`	Cubic anisotropy directions
`Kc1, Kc2, Kc3`	Cubic anisotropy constants (J/m3)
`Msat`	Saturation magnetization (A/m)

Output Quantities	
`B_anis`	Anisotropy field (T)
`Edens_anis`	Total anisotropy energy density (including DMI) (J/m3)
`E_anis`	Total anisotropy energy (including DMI) (J)

DMI field

Interfacially-induced DMI energy density

$$\varepsilon = D\,[m_z(\nabla \cdot \boldsymbol{m}) - (\boldsymbol{m} \cdot \nabla)m_z]$$

Bulk DMI energy density

$$\varepsilon = D\,\boldsymbol{m} \cdot (\nabla \times \boldsymbol{m})$$

Note:

Only single DMI type allowed at once

Regional Material Parameters	
`Dind`	Interfacially-induced DMI strength (J/m2)
`Dbulk`	Bulk DMI strength (J/m2)
`Msat`	Saturation magnetization (A/m)

Output Quantities	
`B_exch`	Exchange field (including DMI) (T)
`Edens_exch`	Total exchange energy density (including DMI) (J/m3)
`E_exch`	Total exchange energy (including DMI) (J)

Other functionalities	
`Ext_InterDind`	Sets Dind coupling between two regions
`Ext_ScaleDind`	Re-scales Dind coupling between two regions
`OpenBC`	Use open boundary conditions (default=false, use Neumann BC). This setting is only relevant for DMI.

External field

Zeeman energy density

$$\varepsilon = -M_s \boldsymbol{m} \cdot \boldsymbol{B}_{ext}$$

Regional Material Parameters	
`Msat`	Saturation magnetization (A/m)
Excitation	
`B_ext`	Externally applied field(T)

Output Quantities	
`B_ext`	Externally applied field(T)
`Edens_zeeman`	Zeeman energy density (J/m3)
`E_zeeman`	Zeeman energy (J)

Thermal field

$$\langle \boldsymbol{H}_{th}(\boldsymbol{r},t)\rangle = 0$$

$$\langle \boldsymbol{H}_{th}(\boldsymbol{r},t), \boldsymbol{H}_{th}(\boldsymbol{r}',t')\rangle = \frac{2k_B T\alpha}{M_s\gamma}\delta(\boldsymbol{r}-\boldsymbol{r}')\delta(t-t')$$

Regional Material Parameters	
`Temp`	Temperature (K)
`alpha`	Damping parameter
`Msat`	Saturation magnetization (A/m)

Output Quantities	
`B_therm`	Thermal field (T)
`Edens_therm`	Thermal energy density (J/m3)
`E_therm`	Thermal energy (J)

Other functionalities	
`ThermSeed`	Sets random seed for thermal noise

Custom fields

Output Quantities	
`B_custom`	User-defined field (T)
`Edens_custom`	Total energy density of user-defined field (J/m3)
`E_custom`	Total energy of user-defined field (J)

Other functionalities	
`AddEdensTerm`	Add a custom energy density term
`AddFieldTerm`	Add a custom effective field term
`RemoveCustomFields`	Remove all custom fields

Zhang-Li STT

Zhang-Li spin-transfer torque

$$\boldsymbol{\tau} = \frac{1+\xi\alpha}{1+\alpha^2}\boldsymbol{m}\times(\boldsymbol{m}\times(\boldsymbol{u}\cdot\nabla)\mathbf{m}) + \frac{\xi-\alpha}{1+\alpha^2}\boldsymbol{m}\times(\boldsymbol{u}\cdot\nabla)\boldsymbol{m}$$

with

$$\boldsymbol{u} = \frac{\mu_B P}{2e\gamma_0 M_s(1+\xi^2)}\boldsymbol{j}$$

Regional Material Parameters	
`Pol`	Electrical current polarization
`xi`	Non-adiabaticity of spin-transfer-torque
`alpha`	Damping parameter
`Msat`	Saturation magnetization (A/m)
Excitation	
`J`	Electrical current density (A/m2)

Output Quantities	
`STTorque`	Spin-transfer torque/γ0 (T)

Other functionalities	
`DisableZhangLiTorque`	Disable Zhang-Li torque (default=false)

Slonczewsi STT

Slonczewski spin-transfer torque

$$\boldsymbol{\tau} = \beta \frac{\epsilon + \alpha\epsilon'}{1+\alpha^2} \boldsymbol{m} \times (\boldsymbol{m}_P \times \mathbf{m}) - \beta \frac{\epsilon' - \alpha\epsilon}{1+\alpha^2} \boldsymbol{m} \times \boldsymbol{m}_P$$

with

$$\beta = \frac{j_z \hbar}{M_s e\, d}$$

$$\epsilon = \frac{P\Lambda^2}{(\Lambda^2+1) + (\Lambda^2-1)(\boldsymbol{m} \cdot \boldsymbol{m}_P)}$$

Regional Material Parameters	
`Pol`	Electrical current polarization
`Lambda`	Slonczewski Λ parameter
`EpsilonPrime`	Slonczewski secondairy STT term ε'
`alpha`	Damping parameter
`Msat`	Saturation magnetization (A/m)
`FreeLayerThickness`	Slonczewski free layer thickness (if set to zero (default), then the thickness will be deduced from the mesh size) (m)
Excitation	
`J`	Electrical current density (A/m2)
`FixedLayer`	Slonczewski fixed layer polarization

Output Quantities	
`STTorque`	Spin-transfer torque/γ0 (T)

Other functionalities	
`DisableSlonczewskiTorque`	Disables Slonczewski torque (default=false)
`FixedLayerPosition`	Position of the fixed layer: FIXEDLAYER_TOP, FIXEDLAYER_BOTTOM (default=FIXEDLAYER_TOP)

GHENT UNIVERSITY

Metis basics

- Login node: metis.niu.edu
- Documentation: http://crcd.niu.edu
- Class directory: /lstr/sahara/phys790a/ -- everybody can write data here
- Environment: use the "`module`" command, available packages: "`module av`", load "`module load <name>`"; Important packages
  - gcc/gcc-14.2.0
  - cuda/cuda-12.8
  - openmpi/openmpi-5.0.7-gcc-14.2.0-cuda-12.8
  - python/python-3.13.5
  - (paraview not available – yet)
- Queuing system torque/PBS: "`qsub`", "`qstat`" (or "`shownodes`"), "`qdel`"
  - Compile code (using the appropriate environment)
  - Write PBS script, containing the environment package loads and code call
  - Submit job with qsub
  - Use interactive job for short jobs:

    ```
    qsub -I -l select=1:ncpus=1:mpiprocs=1:ngpus=1:mem=16gb,walltime=00:15:00 -j oe
    ```

- Job monitor: https://crcd.niu.edu/crcd/crcd-at-work/qstat-monitor.shtml

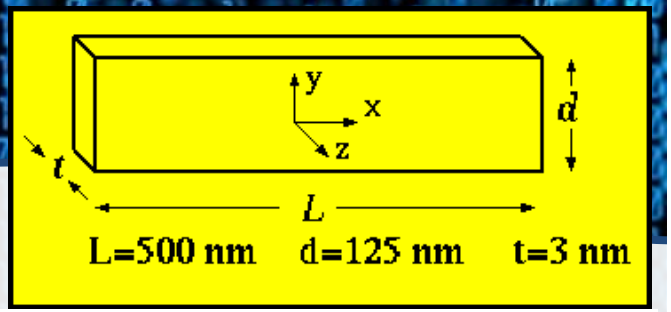
Mumax3 on metis

Mumax3 is installed globally on metis

mumax setup/run use (both interactively and in batch scripts):

```
module load mumax/mumax-3.11.1-cuda-12.6
mumax3 <script name>
```

Example



NIST example:
https://www.ctcms.nist.gov/~rdm/std4/spec4.html

Geometry of film

- thickness, t=3 nm,
- length, L=500 nm, and
- width, d=125 nm

Material parameters (~ permalloy)

- A = 1.3e-11 J/m
- M_s = 8.0e5 A/m

} exchange length: 5.68 nm

- K = 0.0
- α = 0.02

Field

- $\mu_0 H_x$=-24.6 mT
- $\mu_0 H_y$= 4.3 mT
- $\mu_0 H_z$= 0.0 mT

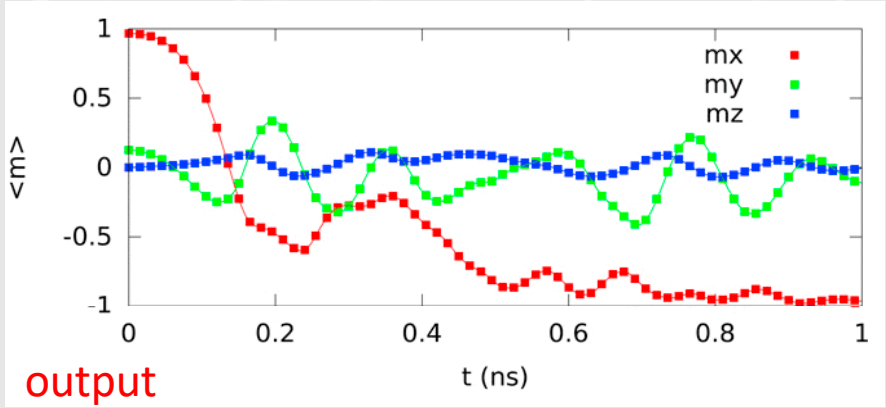
Format conversion, see for formats:

```
mumax3-convert -help
```

Visualization: mumax-view, paraview, …

script

```
SetGridsize(128, 32, 1)
SetCellsize(500e-9/128, 125e-9/32, 3e-9)

Msat  = 800e3
Aex   = 13e-12
alpha = 0.02

m = uniform(1, .1, 0)
relax()
save(m)    // relaxed state

autosave(m, 200e-12)
tableautosave(10e-12)

B_ext = vector(-24.6E-3, 4.3E-3, 0)
run(1e-9)
```



output

Further Resources

- Mumax^3:
  - https://mumax.github.io/
- Mumax3 workshop
  - https://mumax.ugent.be/mumax3-workshop/
- Publications
  - Original paper: https://doi.org/10.1063/1.4899186 “The design and verification of MuMax3”
  - “Tutorial: Simulating modern magnetic material systems in mumax3”: https://doi.org/10.1063/5.0160988

Requirements: machine with CUDA enabled GPU, use metis or google CoLab