mumax^3

GPU-accelerated micromagnetism

---

mumax 3.10 examples

These are example input scripts, the full API can be found here.

mumax^3 input files are run with the command

```
mumax3 myfile.mx3
```

Output is automatically stored in the "myfile.out" directory. Additionally, a web interface provides live output. Default is `http://localhost:35367`.
For more details, run `mumax3 -help` which will show the available command-line flags (e.g. to select a certain GPU).

Getting started with Standard Problem #4

Let's start with the classic mumag standard problem 4, as defined here.

```
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)
```

This example should be pretty straight-forward to follow. Space-dependent output is stored in OVF format, which is compatible with OOMMF and can be converted with `mumax3-convert`. Below is the output converted to PNG.

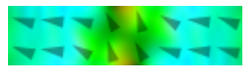
The data table is stored in a simple text format compatible with gnuplot, like used for the plot below.

output


m000000


m000001

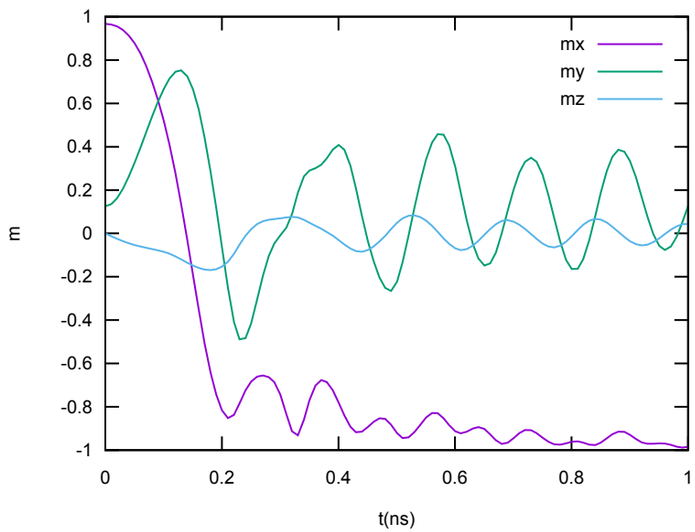

m000002


m000003


m000004


m000005


m000006



m.svg

Standard Problem #2

Using the scripting language explained above, relatively complex input files can be easily defined. E.g. micromagnetic standard problem #2 specifies the simulation size in exchange lengths. The script below calculates the exchange length and chooses cells not larger than 0.75 exchange lengths so that the number of cells is a power of two (for best performance).

```
Msat  = 1000e3
Aex   = 10e-12

// define exchange length
lex := sqrt(10e-12 / (0.5 * mu0 * pow(1000e3 ,2)))
```

```
d     := 30 * lex                       // we test for d/lex = 30
Sizex := 5*d                            // magnet size x
Sizey := 1*d
Sizez := 0.1*d

nx := pow(2, ilogb(Sizex / (0.75*lex)))  // power-of-two number of cells
ny := pow(2, ilogb(Sizey / (0.75*lex)))  // not larger than 0.75 exchange lengths

SetGridSize(nx, ny, 1)
SetCellSize(Sizex/nx, Sizey/ny, Sizez)

m = Uniform(1, 0.1, 0)                  // initial mag
relax()

save(m)                                 // remanent magnetization
print("<m> for d/lex=30: ", m.average())
```

output

m000000

This example saves and prints the remanent magnetization state so we can verify it against known values.

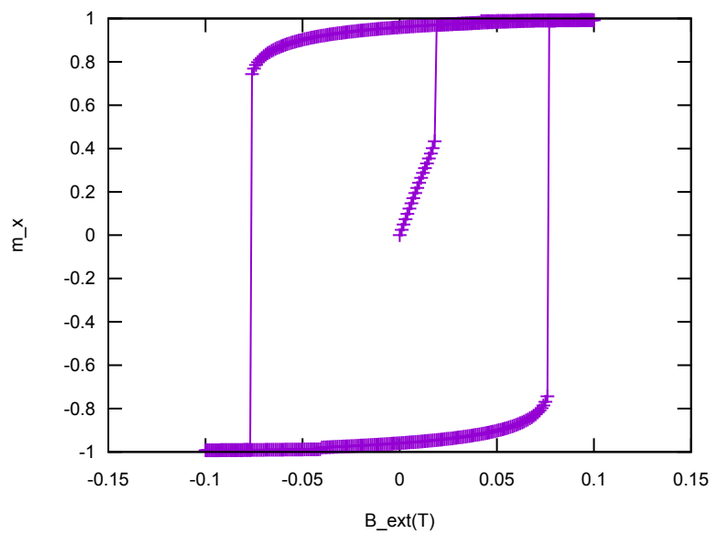
Hysteresis

Below is an example of a hysteresis loop where we step the applied field in small increments and find the magnetization ground state after each step. Minimize() finds the ground state using the conjugate gradient method, which is very fast. However, this method might fail on very high energy initial states like a random magnetization. In that case, Relax() is more robust (albeit much slower).

```
SetGridsize(128, 32, 1)
SetCellsize(4e-9, 4e-9, 30e-9)

Msat  = 800e3
Aex   = 13e-12

m = randomMag()
relax()         // high-energy states best minimized by relax()


Bmax  := 100.0e-3
Bstep :=  1.0e-3
MinimizerStop = 1e-6
TableAdd(B_ext)

for B:=0.0; B<=Bmax; B+=Bstep{
    B_ext = vector(B, 0, 0)
    minimize()   // small changes best minimized by minimize()
```

```
    tablesave()
}

for B:=Bmax; B>=-Bmax; B-=Bstep{
    B_ext = vector(B, 0, 0)
    minimize()   // small changes best minimized by minimize()
    tablesave()
}

for B:=-Bmax; B<=Bmax; B+=Bstep{
    B_ext = vector(B, 0, 0)
    minimize()   // small changes best minimized by minimize()
    tablesave()
}
```

output



Geometry

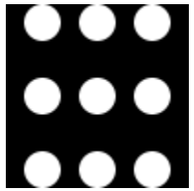
mumax3 has powerful API to programatically define geometries. A number of primitive shapes are defined, like ellipses, rectangles, etc. They can be transformed (rotated, translated) and combined using boolean logic (add, sub, inverse). All positions are specified in meters and the origin lies in the center of the simulation box. See the full API. Edges can be smoothed to reduce staircase effects. `EdgeSmooth=n` means n^3 samples per cell are used to determine its volume. `EdgeSmooth=0` implies a staircase approximation, while `EdgeSmooth=8` results in quite accurately resolved edges.

```
SetGridsize(100, 100, 50)
SetCellsize(1e-6/100, 1e-6/100, 1e-6/50)

EdgeSmooth = 8
```

```
setgeom( rect(800e-9, 500e-9) )
saveas(geom, "rect")

setgeom( cylinder(800e-9, inf) )
saveas(geom, "cylinder")

setgeom( circle(200e-9).repeat(300e-9, 400e-9, 0) )
saveas(geom, "circle_repeat")

setgeom( cylinder(800e-9, inf).inverse() )
saveas(geom, "cylinder_inverse")

setgeom( cylinder(800e-9, 600e-9).transl(200e-9, 100e-9, 0) )
saveas(geom, "cylinder_transl")

setgeom( ellipsoid(800e-9, 600e-9, 500e-9) )
saveas(geom, "ellipsoid")

setgeom( cuboid(800e-9, 600e-9, 500e-9) )
saveas(geom, "cuboid")

setgeom( cuboid(800e-9, 600e-9, 500e-9).rotz(-10*pi/180) )
saveas(geom, "cuboid_rotZ")

setgeom( layers(0, 25) )
saveas(geom, "layers")

setgeom( cell(50, 20, 0) )
saveas(geom, "cell")

setgeom( xrange(0, inf) )
saveas(geom, "xrange")

a := cylinder(600e-9, 600e-9).transl(-150e-9, 50e-9, 0 )
b := rect(600e-9, 600e-9).transl(150e-9, -50e-9, 0)

setgeom( a.add(b) )
saveas(geom, "logicAdd")

setgeom( a.sub(b) )
saveas(geom, "logicSub")

setgeom( a.intersect(b) )
saveas(geom, "logicAnd")

setgeom( a.xor(b) )
saveas(geom, "logicXor")

setgeom( imageShape("mask.png") )
saveas(geom, "imageShape")
```

output


cell


circle_repeat

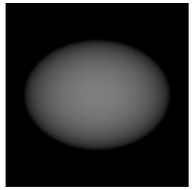

cuboid


cuboid_rotZ


cylinder


cylinder_inverse


cylinder_transl


ellipsoid

imageShape


layers


logicAdd


logicAnd


logicSub


logicXor


rect


xrange

Note: these are 3D geometries seen from above. The displayed cell filling is averaged along the thickness (notable in ellipse and layers example). Black means empty space, white is filled.

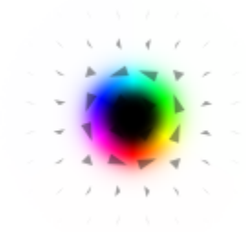
Initial Magnetization

Some initial magnetization functions are provided, as well as transformations similar to those on Shapes. See the Config API.

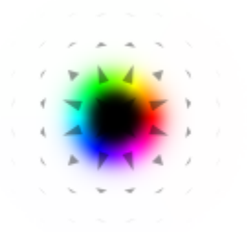
```
setgridsize(256, 128, 1)
setcellsize(5e-9, 5e-9, 5e-9)

m = Uniform(1, 1, 0)  // no need to normalize length
saveas(m, "uniform")

m = Vortex(1, -1)     // circulation, polarization
saveas(m, "vortex")

m = TwoDomain(1,0,0,  0,1,0,  -1,0,0) // Néel wall
saveas(m, "twodomain")

m = RandomMag()
saveas(m, "randommag")

m = TwoDomain(1,0,0,  0,1,0,  -1,0,0).rotz(-pi/4)
saveas(m, "twodomain_rot")
```

```
m = VortexWall(1, -1, 1, 1)
saveas(m, "vortexwall")

m = VortexWall(1, -1, 1, 1).scale(1/2, 1, 1)
saveas(m, "vortexwall_scale")

m = Vortex(1,-1).transl(100e-9, 50e-9, 0)
saveas(m, "vortex_transl")

m = Vortex(1,-1).Add(0.1, randomMag())
saveas(m, "vortex_add_random")

m = BlochSkyrmion(1, -1).scale(3,3,1)
saveas(m, "Bloch_skyrmion")

m = NeelSkyrmion(1,-1).scale(3,3,1)
saveas(m, "Néel_skyrmion")

// set m in only a part of space, or a single cell:
m = uniform(1, 1, 1)
m.setInShape(cylinder(400e-9, 100e-9), vortex(1, -1))
m.setCell(20, 10, 0, vector(0.1, 0.1, -0.9)) // set in cell index  [20,10,0]
saveas(m, "setInShape_setCell")

//Read m form .ovf file.
m.loadfile("myfile.ovf")
saveas(m, "loadfile")
```

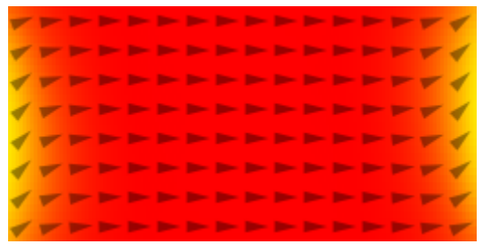
output



Bloch_skyrmion

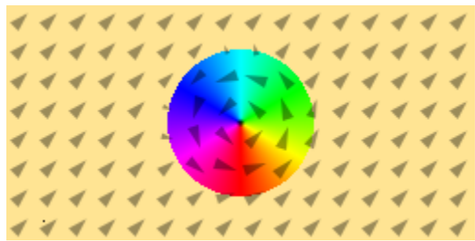


Néel_skyrmion

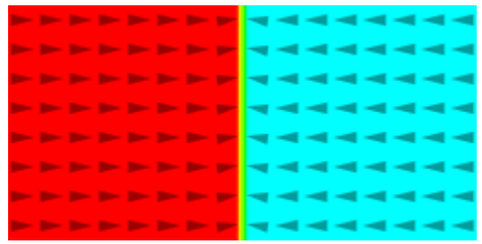


loadfile

randommag



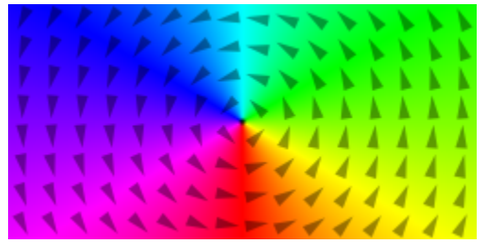
setInShape_setCell

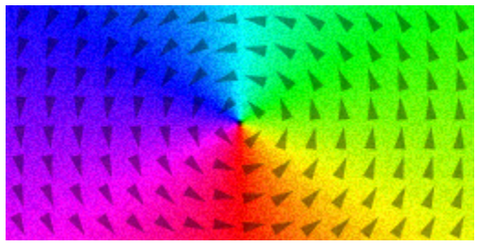


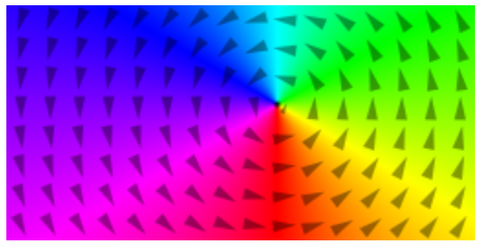
twodomain

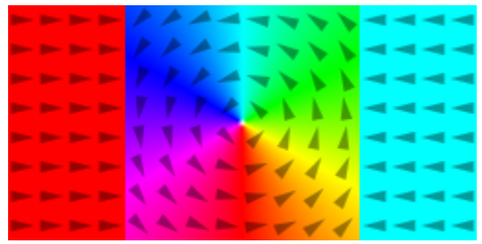

twodomain_rot

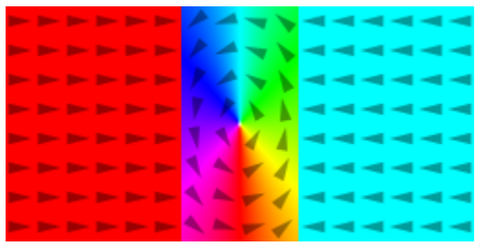

uniform


vortex


vortex_add_random


vortex_transl


vortexwall


vortexwall_scale

These initial states are approximate, after setting them it is a good idea to relax the magnetization to the actual ground state. The magnetization can also be set in separate regions, see below.

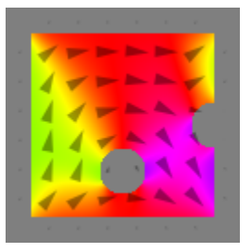
Interlude: Rotating Cheese

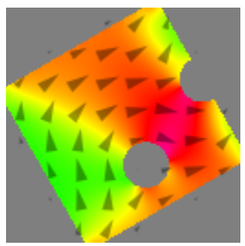
In this example we define a geometry that looks like a slice of cheese and have it rotate in time.

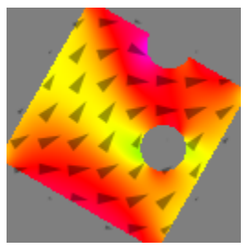
```
setgridsize(128, 128, 1)
setcellsize(2e-9, 2e-9, 2e-9)

d      := 200e-9
sq     := rect(d, d)                // square with side d

h     := 50e-9
hole  := cylinder(h, h)             // circle with diameter h
hole1 := hole.transl(100e-9, 0, 0)  // translated circle #1
hole2 := hole.transl(0, -50e-9, 0)  // translated cricle #2
cheese:= sq.sub(hole1).sub(hole2)// subtract the circles from the square (makes holes).
setgeom(cheese)
```

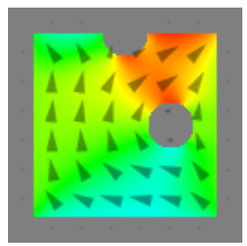
```
msat = 600e3
aex = 12e-13
alpha = 3

// rotate the cheese.
for i:=0; i<=90; i=i+30{
        angle := i*pi/180
        setgeom(cheese.rotz(angle))
        m = uniform(cos(angle), sin(angle), 0)
        minimize()
        save(m)
}
```

output


m000000


m000001


m000002


m000003

Regions: Space-dependent Parameters

Space-dependent parameters are defined using material *regions*. Regions are numbered 0-255 and represent different materials. Each cell can belong to only one region. At the start of a simulation all cells have region number 0.

Regions are defined with `defregion(number, shape)`, where `shape` is explained in the geometry example.

When you're not using regions, like in the above examples, you'll probably set parameters with a simple assign:

```
Aex = 12e-13
```

Behind the screens, this sets Aex in *all* regions.

It's always a good idea to output the `regions` quantity, as well as all your material parameters.

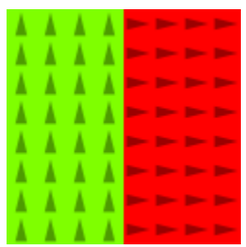
```
N := 128
setgridsize(N, N, 1)
c := 4e-9
setcellsize(c, c, c)
```

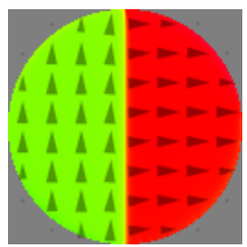
```
// disk with different anisotropy in left and right half
setgeom(circle(N*c))
defregion(1, xrange(0, inf))  // left half
defregion(2, xrange(-inf, 0)) // right half
save(regions)

Ku1.setregion(1, .1e6)
anisU.setRegion(1, vector(1, 0, 0))

Ku1.setregion(2, .2e6)
anisU.setRegion(2, vector(0, 1, 0))

save(Ku1)
save(anisU)

Msat = 800e3 // sets it everywhere
save(Msat)

Aex = 12e-13
alpha = 1

m.setRegion(1, uniform(1, 1, 0))
m.setRegion(2, uniform(-1, 1, 0))
saveas(m, "m_inital")
run(.1e-9)
saveas(m, "m_final")
```

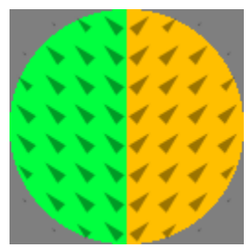
output


Ku1000000


Msat000000


anisU000000


m_final


m_inital


regions000000

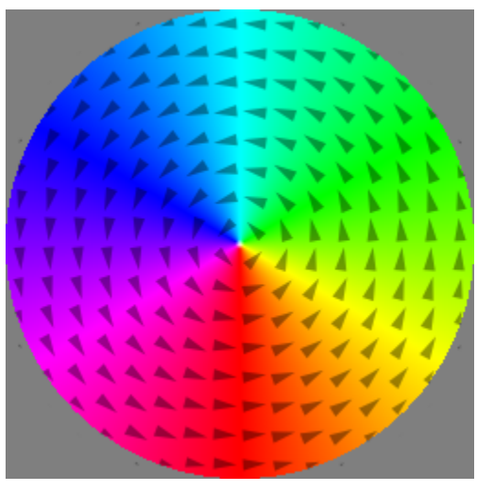
Slicing and dicing output

The example below illustrates how to save only the part of the output you're interested in.

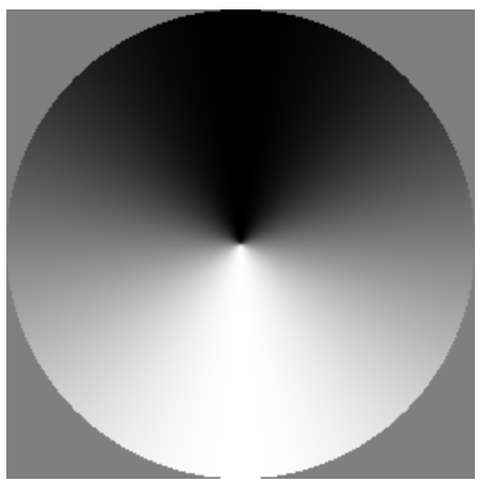
```
Nx := 256
Ny := 256
Nz := 1
setgridsize(Ny, Nx, Nz)
c := 4e-9
```

```
setcellsize(c, c, c)

setgeom(circle(Nx*c))

Msat  = 800e3
Aex   = 12e-13
alpha = 1
m = vortex(1, 1)

save(m)
save(m.Comp(0))
save(Crop(m, 0, Nx/2, 0, Ny/2, 0, Nz))

mx := m.Comp(0)
mx_center := CropY(mx, Ny/4, 3*Ny/4)
save(mx_center)
```

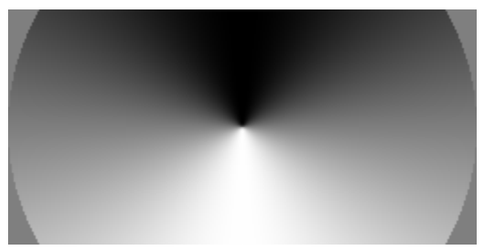
output



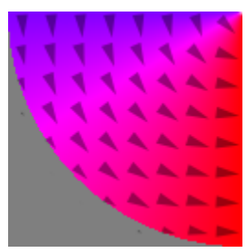
m000000



m_x000000



m_x_yrange64-192_000000



m_xrange0-128_yrange0-128_000000

Magnetic Force Microscopy

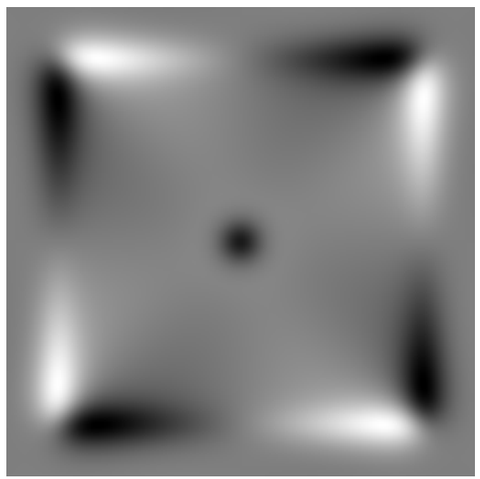
Mumax3 has built-in generation of MFM images from the magnetization. The MFM tip lift can be freely chosen. By default the tip magnetization is modeled as a point monopole at the apex. This is sufficient for most situations. Nevertheless, it is also possible to model partially magnetized tips by setting MFMDipole to the magnetized portion of the tip, in meters. E.g., if only the first 20nm of the tip is (vertically) magnetized, set MFMDipole=20e-9.

```
setgridsize(256, 256, 1)
setcellsize(2e-9, 2e-9, 1e-9)
```

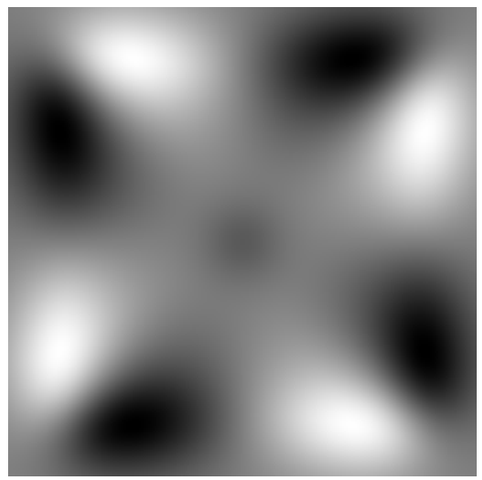
```
setgeom(rect(400e-9, 400e-9))

msat    = 600e3
aex     = 10e-12
m       = vortex(1, 1)

relax()
save(m)

MFMLift = 10e-9
saveas(MFM, "lift_10nm")

MFMLift = 40e-9
saveas(MFM, "lift_40nm")

MFMLift = 90e-9
saveas(MFM, "lift_90nm")
```

output

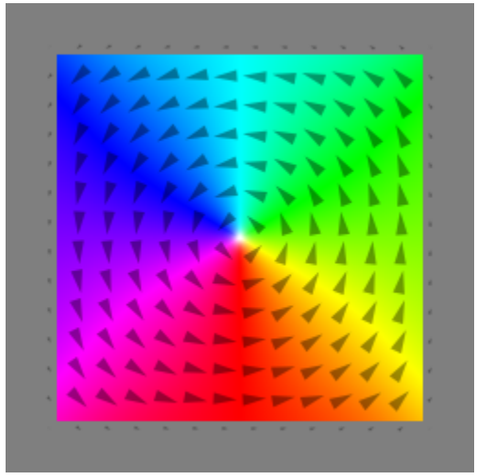
lift_10nm



lift_40nm



lift_90nm



m000000

PMA Racetrack

In this example we drive a domain wall in PMA material by spin-transfer torque. We set up a post-step function that makes the simulation box "follow" the domain wall. Like this, only a small number of cells is needed to simulate an infinitely long magnetic wire.

```
setGridSize(128, 128, 1)
setCellSize(2e-9, 2e-9, 1e-9)

Msat    = 600e3
Aex     = 10e-12
anisU   = vector(0, 0, 1)
Ku1     = 0.59e6
alpha   = 0.02
Xi      = 0.2

m     = twoDomain(0, 0, 1, 1, 1, 0, 0, 0, -1) // up-down domains with wall between Bloch and Néél type
relax()

// Set post-step function that centers simulation window on domain wall.
ext_centerWall(2) // keep m[2] (= m_z) close to zero

// Schedule output
autosave(m, 100e-12)

// Run for 1ns with current through the sample
j   = vector(1.5e13, 0, 0)
pol = 1
run(.5e-9)
```

output

m000000 m000001 m000002 m000003 m000004 m000005

Since we center on the domain wall we can not see that it is actually moving, but the domain wall breakdown is visible.

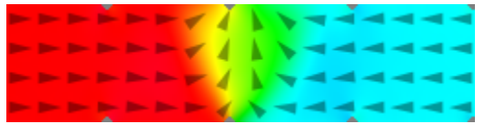
Py Racetrack

In this example we drive a vortex wall in Permalloy by spin-transfer torque. The simulation box "follows" the domain wall. By removing surface charges at the left and right ends, we mimic an infintely long wire.

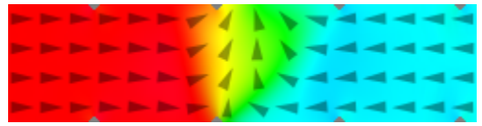
```
setGridSize(256, 64, 1)
setCellSize(3e-9, 3e-9, 10e-9)

Msat    = 860e3
Aex     = 13e-12
Xi      = 0.1
alpha   = 0.02
m       = twodomain(1,0,0,  0,1,0,  -1,0,0)

notches := rect(15e-9, 15e-9).RotZ(45*pi/180).Repeat(200e-9, 64*3e-9, 0).Transl(0, 32*3e-9, 0)
setGeom(notches.inverse())

// Remove surface charges from left (mx=1) and right (mx=-1) sides to mimic infinitely long wire. We have to specify the region (0) at the boundaries.
BoundaryRegion := 0
MagLeft        := 1
MagRight       := -1
ext_rmSurfaceCharge(BoundaryRegion, MagLeft, MagRight)

relax()

ext_centerWall(0) // keep m[0] (m_x) close to zero

// Schedule output
autosave(m, 50e-12)
tableadd(ext_dwpos)   // domain wall position
tableautosave(10e-12)

// Run the simulation with current through the sample
pol = 0.56
J   = vector(-10e12, 0, 0)
Run(0.5e-9)
```

output

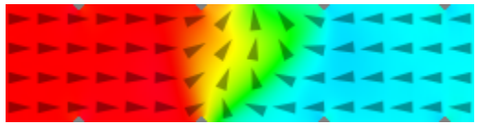
m000000



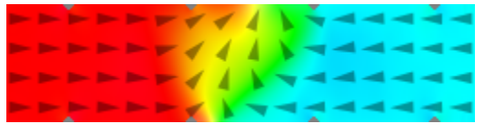
m000001



m000002

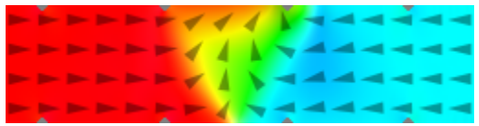


m000003

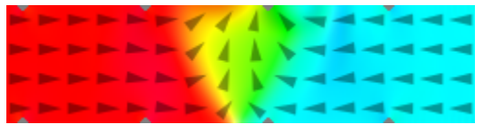


m000004

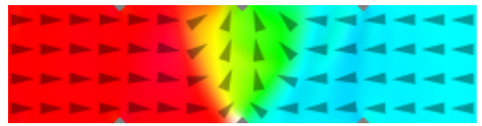
m000005



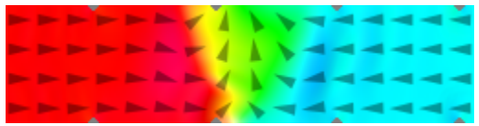
m000006



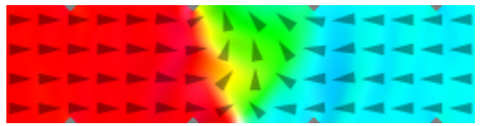
m000007



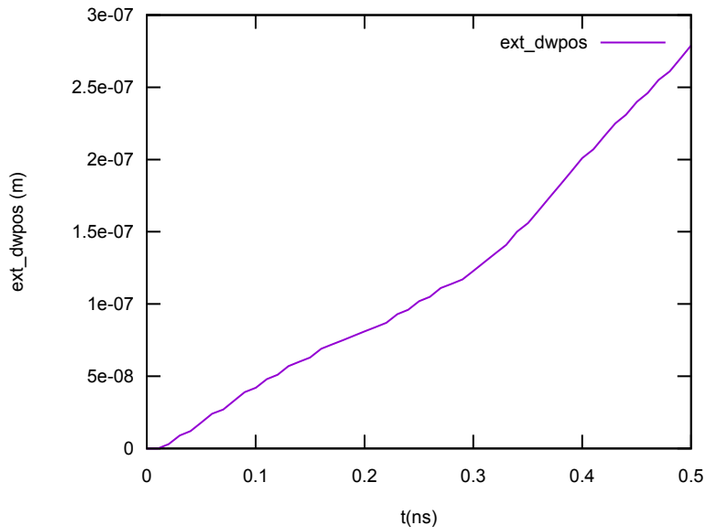
m000008



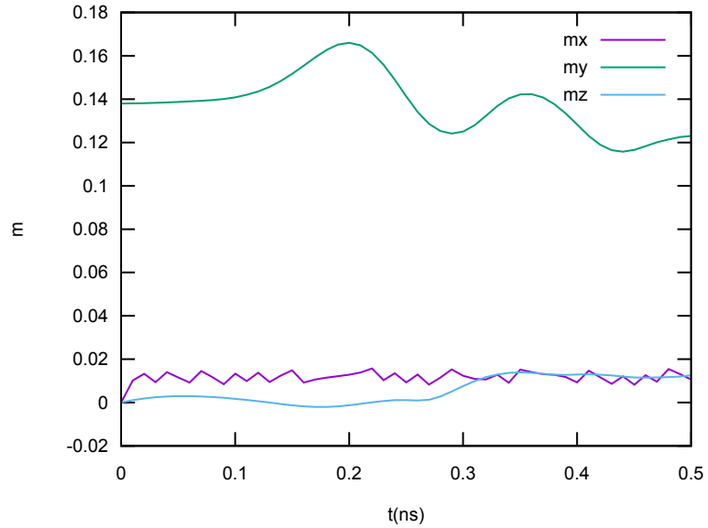
m000009



m000010



ext_dwpos.svg



m.svg

Since we center on the domain wall we can not really see the motion, despite the vortex wall moving pretty fast. Note the absence of closure domains at the edges due to the surface charges being removed there.

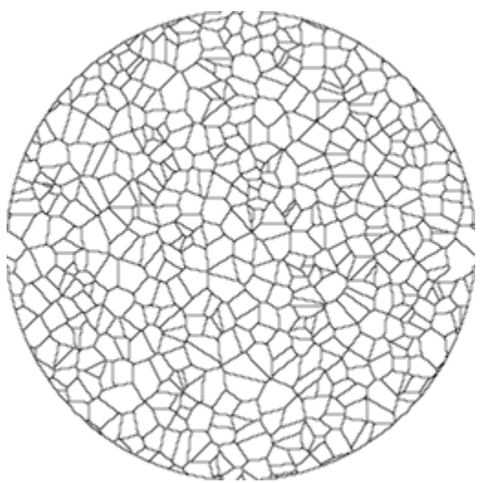
Voronoi tessellation

In this example we use regions to specify grains in a material. The built-in extension `ext_makegrains` is used to define grain-like regions using Voronoi tessellation. We vary the material parameters in each grain.

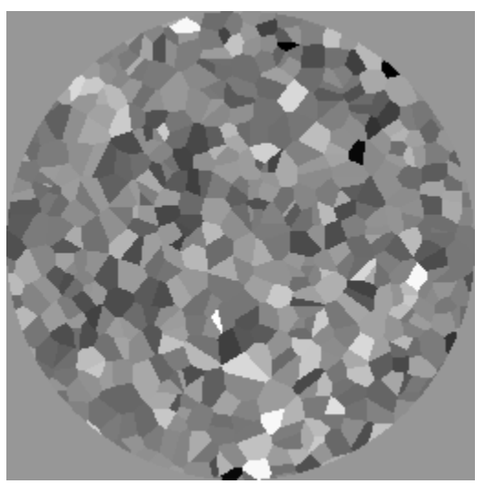
```
N := 256
c := 4e-9
d := 40e-9
setgridsize(N, N, 1)
setcellsize(c, c, d)

setGeom(circle(N*c))

// define grains with region number 0-255
grainSize  := 40e-9  // m
randomSeed := 1234567
maxRegion  := 255
ext_makegrains(grainSize, maxRegion, randomSeed)

defregion(256, circle(N*c).inverse()) // region 256 is outside, not really needed

alpha = 3
Kc1   = 1000
Aex   = 13e-12
Msat  = 860e3
```

```
// set random parameters per region
for i:=0; i<maxRegion; i++{
        // random cubic anisotropy direction
        axis1  := vector(randNorm(), randNorm(), randNorm())
        helper := vector(randNorm(), randNorm(), randNorm())
        axis2  := axis1.cross(helper)  // perpendicular to axis1
        AnisC1.SetRegion(i, axis1)     // axes need not be normalized
        AnisC2.SetRegion(i, axis2)

        // random 10% anisotropy variation
        K := 1e5
        Kc1.SetRegion(i, K + randNorm() * 0.1 * K)
}

// reduce exchange coupling between grains by 10%
for i:=0; i<maxRegion; i++{
        for j:=i+1; j<maxRegion; j++{
                ext_ScaleExchange(i, j, 0.9)
        }
}

m = vortex(1, 1)
run(.1e-9)

save(regions)
save(Kc1)
save(AnisC1)
save(AnisC2)
save(m)
save(exchCoupling)
```

output

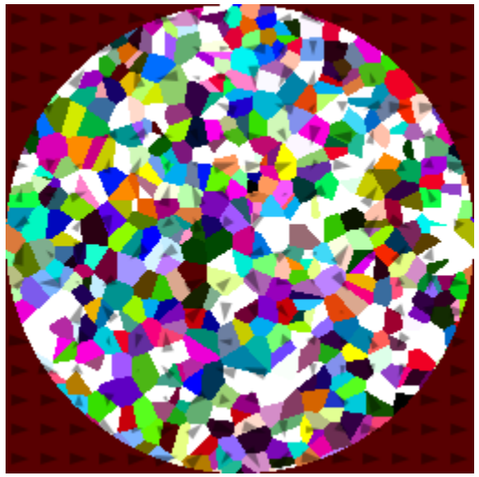


ExchCoupling000000

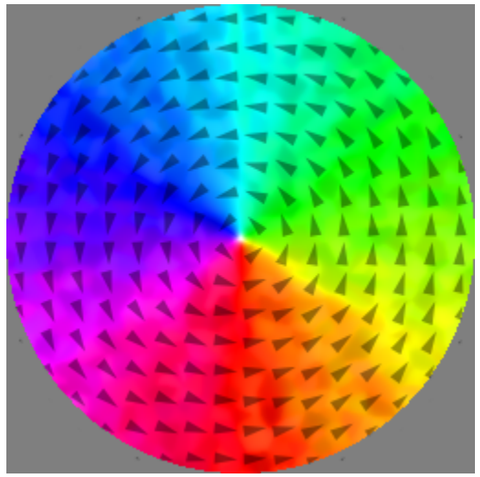


Kc1000000

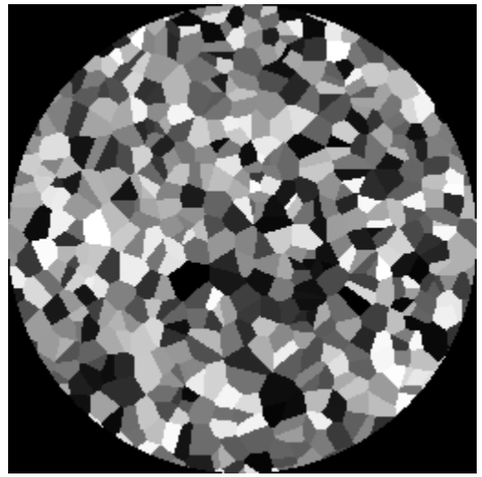
anisC1000000



anisC2000000



m000000



regions000000

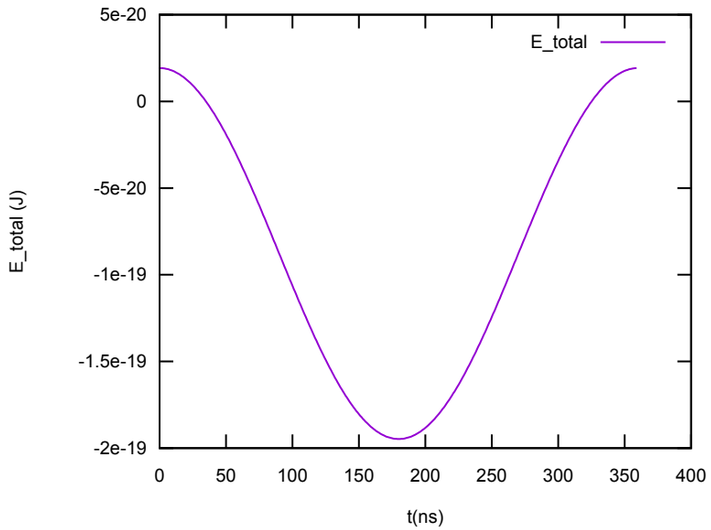
RKKY

Scaling the exchange coupling between regions can be used to obtain antiferromagnetic coupling like the RKKY interaction. In that case we only model the magnetic layers and do not explicitly add a spacer layer (which is negligibly thin). We scale the exchange coupling to get the desired RKKY strength: `scale = (RKKY * cellsize_z) / (2 * Aex)`.

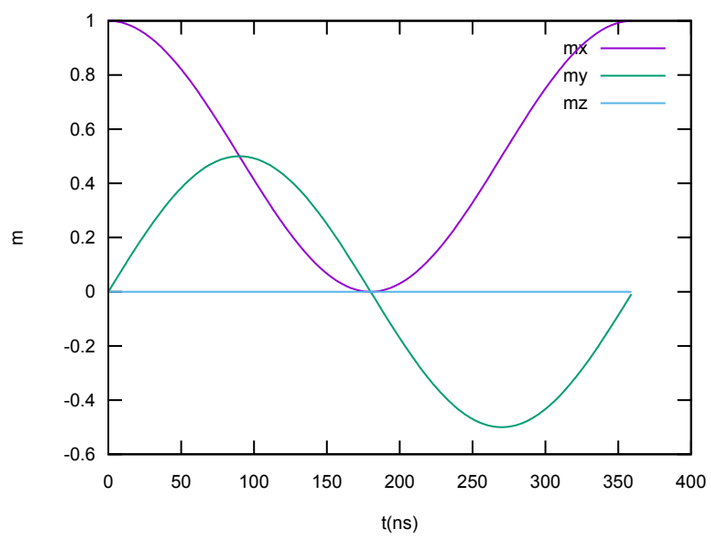
```
N := 10
setgridsize(N, N, 2)

c := 1e-9
setcellsize(c, c, c)

defRegion(0, layer(0))
defRegion(1, layer(1))

Msat = 1e6

Aex  = 10e-12
RKKY := -1e-3 // 1mJ/m2
scale := (RKKY * c) / (2 * Aex.Average())
ext_scaleExchange(0, 1, scale)

tableAdd(E_total)

m.setRegion(0, uniform(1, 0, 0))

for ang:=0; ang<360; ang++{
        m.setRegion(1, uniform(cos(ang*pi/180), sin(ang*pi/180), 0))
        t = ang * 1e-9 // output "time" is really angle
```

```
        tablesave()
}
```

output



E_total.svg



m.svg

Slonczewski STT

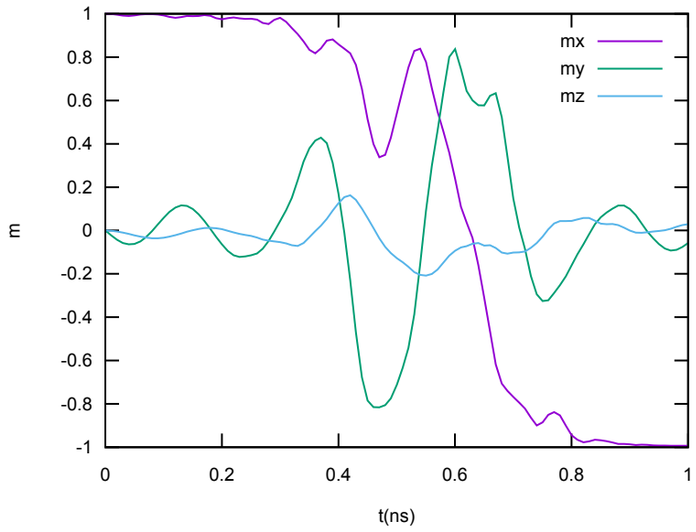
Example of a spin-torque MRAM stack consisting of a fixed layer, spacer and free layer. Only the free layer magnetization is explicitly modeled, so we use a 2D grid. The fixed layer polarization is set with `FixedLayer = ...`, which can be space-dependent. The spacer layer properties are modeled by setting the parameters `Lambda` and `EpsilonPrime`. Finally `Pol` sets the current polarization and `J` the current density, which should be along z in this case. Below we switch an MRAM bit.

```
// geometry
sizeX := 160e-9
sizeY := 80e-9
sizeZ := 5e-9

Nx := 64
Ny := 32

setgridsize(Nx, Ny, 1)
setcellsize(sizeX/Nx, sizeY/Ny, sizeZ)
setGeom(ellipse(sizeX, sizeY))

// set up free layer
Msat  = 800e3
Aex   = 13e-12
alpha = 0.01
m     = uniform(1, 0, 0)

// set up spacer layer parameters
lambda       = 1
Pol          = 0.5669
epsilonprime = 0

// set up fixed layer polarization
angle := 20
px := cos(angle * pi/180)
py := sin(angle * pi/180)
fixedlayer = vector(px, py, 0)

// send current
Jtot := -0.008            // total current in A
area := sizeX*sizeY*pi/4
jc   := Jtot / area       // current density in A/m2
J = vector(0, 0, jc)

// schedule output & run
autosave(m, 100e-12)
tableautosave(10e-12)
run(1e-9)
```

output

m000000 m000001 m000002 m000003 m000004 m000005 m000006 m000007 m000008


m000009 m000010



m.svg

Spinning hard disk

Using the `Shift` function, we can shift the system (magnetization, regions and geometry) by a given number of cells. Here we use this feature to simulate a moving hard disk platter. A time-dependent gaussian field profile mimics the write field.

```
Nx := 512
Ny := 128
c := 5e-9
setgridsize(Nx, Ny, 1)
setcellsize(c, c, c)

ext_makegrains(30e-9, 256, 0)

// PMA material
Ku1   = 0.4e6
Aex   = 10e-12
Msat  = 600e3
alpha = 1

delta := 0.2 // anisotropy variation

for i:=0; i<256; i++{
```

```
        // random cubic anisotropy direction
        AnisU.SetRegion(i, vector(delta*(rand()-0.5), delta*(rand()-0.5), 1))

        // strongly reduce exchange coupling between grains
        for j:=i+1; j<256; j++{
                ext_scaleExchange(i, j, 0.1)
        }
}

m = uniform(0, 0, 1)

// Gaussian external field profile
mask := newVectorMask(Nx, Ny, 1)
for i:=0; i<Nx; i++{
        for j:=0; j<Ny; j++{
                r := index2coord(i, j, 0)
                x := r.X()
                y := r.Y()
                Bz := exp(-pow((x-500e-9)/100e-9, 2)) * exp(-pow(y/250e-9, 2))
                mask.setVector(i, j, 0, vector(0, 0, Bz))
        }
}

// 500Mbit/s oscillting write field
f := 0.5e9
A := 1.5
B_ext.add(mask, -A*sin(2*pi*f*t))

autosave(m, 600e-12)

// Spin the hard disk
ShiftMagR = vector(0, 0, 1) // new magnetization to enter
for i:=0; i<120; i++{
        run(30e-12)
        Shift(-1) // one cell to the left
}
```

output

m000000

m000001

m000002

m000003

m000004

m000005

m000006

mumax^3

GPU-accelerated micromagnetism

abs(float64) float64

Abs returns the absolute value of x.

acos(float64) float64

Acos returns the arccosine, in radians, of x.

acosh(float64) float64

Acosh returns the inverse hyperbolic cosine of x.

Add(Quantity, Quantity) Quantity

Add two quantities

methods: EvalTo( )

examples: [3] [4] [5] [11] [13] [15]

AddEdensTerm(Quantity)

Add an expression to Edens.

AddFieldTerm(Quantity)

Add an expression to B_eff.

Aex

Misc

Go to mumax3.9c API

Exchange stiffness (J/m)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [1] [2] [3] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

alpha

Landau-Lifshitz damping constant

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [1] [6] [7] [8] [10] [11] [12] [14] [15]

anisC1

Cubic anisotropy direction #1

methods: Average( ) Comp(int) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [12]

anisC2

Cubic anisotorpy directon #2

methods: Average( ) Comp(int) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [12]

anisU

Uniaxial anisotropy direction

methods: Average( ) Comp(int) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [7] [10] [15]

Antivortex(int, int) Config

Antivortex magnetization with given circulation and core polarization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

asin(float64) float64

Asin returns the arcsine, in radians, of x.

asinh(float64) float64

Asinh returns the inverse hyperbolic sine of x.

atan(float64) float64

Atan returns the arctangent, in radians, of x.

atan2(float64, float64) float64

Atan2 returns the arc tangent of y/x, using the signs of the two to determine the quadrant of the return value.

atanh(float64) float64

Atanh returns the inverse hyperbolic tangent of x.

AutoSave(Quantity, float64)

Auto save space-dependent quantity every period (s).

examples: [1] [10] [11] [14] [15]

AutoSnapshot(Quantity, float64)

Auto save image of quantity every period (s).

B1

First magneto-elastic coupling constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

B2

Second magneto-elastic coupling constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

B_anis

Anisotropy field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_custom

User-defined field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_demag

Magnetostatic field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_eff

Effective field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_exch

Exchange field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_ext

Externally applied field (T)

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(data.Vector) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [1] [3] [15]

B_mel

Magneto-elastic filed (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_therm

Thermal field (T)

methods: AddTo(Slice) EvalTo(Slice)

BlochSkyrmion(int, int) Config

Bloch skyrmion magnetization with given chirality and core polarization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [5]

cbrt(float64) float64

Cbrt returns the cube root of x.

ceil(float64) float64

Ceil returns the least integer value greater than or equal to x.

Cell(int, int, int) Shape

Single cell with given integer index (i, j, k)

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(

float64 float64 float64) Xor(Shape)

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

Circle(float64) Shape

2D Circle with diameter in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4] [6] [7] [8] [12]

Cone(float64, float64) Shape

3D Cone with diameter and height in meter. The top of the cone points in the +z direction.

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

Conical(data.Vector, data.Vector, float64) Config

Conical state for given wave vector, cone direction, and cone angle

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

Const(float64) Quantity

Constant, uniform number

methods: EvalTo( )

ConstVector(float64, float64, float64) Quantity

Constant, uniform vector

methods: EvalTo( )

cos(float64) float64

Cos returns the cosine of the radian argument x.

examples: [6] [13] [14]

`cosh(float64) float64`

Cosh returns the hyperbolic cosine of x.

`Crop(Quantity, int, int, int, int, int, int) *cropped`

Crops a quantity to cell ranges [x1,x2[, [y1,y2[, [z1,z2[

methods: Average( ) EvalTo(Slice)

examples: [8]

`CropLayer(Quantity, int) *cropped`

Crops a quantity to a single layer

methods: Average( ) EvalTo(Slice)

`CropRegion(Quantity, int) *cropped`

Crops a quantity to a region

methods: Average( ) EvalTo(Slice)

`CropX(Quantity, int, int) *cropped`

Crops a quantity to cell ranges [x1,x2[

methods: Average( ) EvalTo(Slice)

`CropY(Quantity, int, int) *cropped`

Crops a quantity to cell ranges [y1,y2[

methods: Average( ) EvalTo(Slice)

examples: [8]

`CropZ(Quantity, int, int) *cropped`

Crops a quantity to cell ranges [z1,z2[

methods: Average( ) EvalTo(Slice)

Cross(Quantity, Quantity) Quantity

Cross product of two vector quantities

methods: EvalTo( )

examples: [12]

Cuboid(float64, float64, float64) Shape

Cuboid with sides in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4]

Cylinder(float64, float64) Shape

3D Cylinder with diameter and height in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4] [5] [6]

Dbulk

Bulk Dzyaloshinskii-Moriya strength (J/m2)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

DefRegion(int, Shape)

Define a material region with given index (0-255) and shape

examples: [7] [12] [13]

DefRegionCell(int, int, int, int)

Set a material region (first argument) in one cell by the index of the cell (last three arguments)

DemagAccuracy

Controls accuracy of demag kernel

Dind

Interfacial Dzyaloshinskii-Moriya strength (J/m2)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

DindCoupling

Average DMI coupling with neighbors (arb.)

methods: Average( ) EvalTo(Slice) Region(int)

DisableSlonczewskiTorque

Disables Slonczewski torque (default=false)

DisableZhangLiTorque

Disables Zhang-Li torque (default=false)

Div(Quantity, Quantity) Quantity

Point-wise division of two quantities

methods: EvalTo( )

DoPrecess

Enables LL precession (default=true)

Dot(Quantity, Quantity) Quantity

Dot product of two vector quantities

methods: EvalTo( )

dt

Time Step (s)

methods: Average( ) EvalTo(Slice) Get( )

DUMP

OutputFormat = DUMP sets text DUMP output

E_anis

total anisotropy energy (J)

methods: Average( ) EvalTo(Slice) Get( )

E_custom

total energy of user-defined field (J)

methods: Average( ) EvalTo(Slice) Get( )

E_demag

Magnetostatic energy (J)

methods: Average( ) EvalTo(Slice) Get( )

E_exch

Total exchange energy (including the DMI energy) (J)

methods: Average( ) EvalTo(Slice) Get( )

E_mel

Magneto-elastic energy (J)

methods: Average( ) EvalTo(Slice) Get( )

E_therm

Thermal energy (J)

methods: Average( ) EvalTo(Slice) Get( )

E_total

total energy (J)

methods: Average( ) EvalTo(Slice) Get( )

examples: [13]

E_Zeeman

Zeeman energy (J)

methods: Average( ) EvalTo(Slice) Get( )

Edens_anis

Anisotropy energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_custom

Energy density of user-defined field. (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_demag

Magnetostatic energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_exch

Total exchange energy density (including the DMI energy density) (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_mel

Magneto-elastic energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_therm

Thermal energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_total

Total energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_Zeeman

Zeeman energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

EdgeSmooth

Geometry edge smoothing with edgeSmooth^3 samples per cell, 0=staircase, ~8=very smooth

examples: [4]

Ellipse(float64, float64) Shape

2D Ellipse with axes in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [14]

Ellipsoid(float64, float64, float64) Shape

3D Ellipsoid with axes in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4]

EnableDemag

Enables/disables demag (default=true)

EpsilonPrime

Slonczewski secondairy STT term ε'

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [14]

erf(float64) float64

Erf returns the error function of x.

erfc(float64) float64

Erfc returns the complementary error function of x.

ExchCoupling

Average exchange coupling with neighbors (arb.)

methods: Average( ) EvalTo(Slice) Region(int)

examples: [12]

Exit()

Exit from the program

exp(float64) float64

Exp returns e**x, the base-e exponential of x.

examples: [15]

exp2(float64) float64

Exp2 returns 2**x, the base-2 exponential of x.

Expect(string, float64, float64, float64)

Used for automated tests: checks if a value is close enough to the expected value

ExpectV(string, data.Vector, data.Vector, float64)

Used for automated tests: checks if a vector is close enough to the expected value

expm1(float64) float64

Expm1 returns e**x - 1, the base-e exponential of x minus 1. It is more accurate than Exp(x) - 1 when x is near zero.

ext_bubbledist

Bubble traveled distance (m)

methods: Average( ) EvalTo(Slice) Get( )

ext_BubbleMz

Center magnetization 1.0 or -1.0 (default = 1.0)

ext_bubblepos

Bubble core position (m)

methods: Average( ) EvalTo(Slice) Get( )

ext_bubblespeed

Bubble velocity (m/s)

methods: Average( ) EvalTo(Slice) Get( )

ext_centerBubble()

centerBubble shifts m after each step to keep the bubble position close to the center of the window

ext_centerWall(int)

centerWall(c) shifts m after each step to keep m_c close to zero

examples: [10] [11]

ext_corepos

Vortex core position (x,y) + polarization (z) (m)

methods: Average( ) EvalTo(Slice) Get( )

ext_dwpos

Position of the simulation window while following a domain wall (m)

methods: Average( ) EvalTo(Slice) Get( )

examples: [11]

ext_dwspeed

Speed of the simulation window while following a domain wall (m/s)

methods: Average( ) EvalTo(Slice) Get( )

ext_dwtilt

PMA domain wall tilt (rad)

methods: Average( ) EvalTo(Slice) Get( )

ext_dwxpos

Position of the simulation window while following a domain wall (m)

methods: Average( ) EvalTo(Slice) Get( )

ext_EnableUnsafe()

Deprecated. Only here to ensure maximal backwards compatibility with mumax3.9c.

ext_InterDind(int, int, float64)

Sets Dind coupling between two regions.

ext_InterExchange(int, int, float64)

Sets exchange coupling between two regions.

ext_make3dgrains(float64, int, int, Shape, int)

3D Voronoi tesselation over shape (grain size, starting region number, num regions, shape, seed)

ext_makegrains(float64, int, int)

Voronoi tesselation (grain size, num regions)

examples: [12] [15]

ext_phi

Azimuthal angle (rad)

methods: Average( ) EvalTo(Slice) Region(int)

ext_rmSurfaceCharge(int, float64, float64)

Compensate magnetic charges on the left and right sides of an in-plane magnetized wire.
Arguments: region, mx on left and right side, resp.

examples: [11]

ext_ScaleDind(int, int, float64)

Re-scales Dind coupling between two regions.

ext_ScaleExchange(int, int, float64)

Re-scales exchange coupling between two regions.

examples: [12] [13] [15]

ext_theta

Polar angle (rad)

methods: Average( ) EvalTo(Slice) Region(int)

ext_topologicalcharge

2D topological charge

methods: Average( ) EvalTo(Slice) Get( )

ext_topologicalchargedensity

2D topological charge density m·(∂m/∂x × ∂m/∂y) (1/m2)

methods: Average( ) EvalTo(Slice) Region(int)

ext_topologicalchargedensitylattice

2D topological charge density according to Berg and Lüscher (1/m2)

methods: Average( ) EvalTo(Slice) Region(int)

ext_topologicalchargelattice

2D topological charge according to Berg and Lüscher

methods: Average( ) EvalTo(Slice) Get( )

exx

exx component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

exy

exy component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

exz

exz component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

eyy

eyy component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

eyz

eyz component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

ezz

ezz component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

F_mel

Magneto-elastic force density (N/m3)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

false

FilenameFormat

printf formatting string for output filenames.

FixDt

Set a fixed time step, 0 disables fixed step (which is the default)

FixedLayer

Slonczewski fixed layer polarization

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(data.Vector) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [14]

FIXEDLAYER_BOTTOM

FixedLayerPosition = FIXEDLAYER_BOTTOM instructs mumax3 that fixed layer is underneath of the free layer

FIXEDLAYER_TOP

FixedLayerPosition = FIXEDLAYER_TOP instructs mumax3 that fixed layer is on top of the free layer

FixedLayerPosition

Position of the fixed layer: FIXEDLAYER_TOP, FIXEDLAYER_BOTTOM (default=FIXEDLAYER_TOP)

floor(float64) float64

Floor returns the greatest integer value less than or equal to x.

Flush()

Flush all pending output to disk.

Fprintln(string, ...interface {})

Print to file

FreeLayerThickness

Slonczewski free layer thickness (if set to zero (default), then the thickness will be deduced from the mesh size) (m)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

frozenspins

Defines spins that should be fixed

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

gamma(float64) float64

Gamma returns the Gamma function of x.

GammaLL

Gyromagnetic ratio in rad/Ts

geom

Cell fill fraction (0..1)

methods: Average( ) EvalTo(Slice) Gpu( )

examples: [4] [6] [7] [8] [9] [11] [12] [14]

GrainRoughness(float64, float64, float64, int) Shape

Grainy surface with different heights per grain with a typical grain size (first argument), minimal height (second argument), and maximal height (third argument). The last argument is a seed for the random number generator.

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

Headroom

Solver headroom (default = 0.8)

heaviside(float64) float64

Returns 1 if x>0, 0 if x<0, and 0.5 if x==0

Helical(data.Vector) Config

Helical state for given wave vector

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

hypot(float64, float64) float64

Hypot returns Sqrt(p*p + q*q), taking care to avoid unnecessary overflow and underflow.

ilogb(float64) int

Ilogb returns the binary exponent of x as an integer.

examples: [2]

ImageShape(string) Shape

Use black/white image as shape

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4]

Index2Coord(int, int, int) data.Vector

Convert cell index to x,y,z coordinate in meter

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

examples: [15]

inf

Inf returns positive infinity if sign >= 0, negative infinity if sign < 0.

examples: [4] [7] [11]

isInf(float64, int) bool

IsInf reports whether f is an infinity, according to sign. If sign > 0, IsInf reports whether f is positive infinity. If sign < 0, IsInf reports

isNaN(float64) bool

IsNaN reports whether f is an IEEE 754 “not-a-number” value.

J

Electrical current density (A/m2)

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(data.Vector) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [10] [11] [12] [13] [14] [15]

j0(float64) float64

J0 returns the order-zero Bessel function of the first kind.

j1(float64) float64

J1 returns the order-one Bessel function of the first kind.

jn(int, float64) float64

Jn returns the order-n Bessel function of the first kind.

Kc1

1st order cubic anisotropy constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [12]

Kc2

2nd order cubic anisotropy constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

Kc3

3rd order cubic anisotropy constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

Ku1

1st order uniaxial anisotropy constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [7] [10] [15]

Ku2

2nd order uniaxial anisotropy constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

Lambda

Slonczewski Λ parameter

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [14]

LastErr

Error of last step

methods: Average( ) EvalTo(Slice) Get( )

Layer(int) Shape

Single layer (along z), by integer index starting from 0

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4] [13] [14]

Layers(int, int) Shape

Part of space between cell layer1 (inclusive) and layer2 (exclusive), in integer indices

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4]

ldexp(float64, int) float64

Ldexp is the inverse of Frexp. It returns frac × 2**exp.

LLtorque

Landau-Lifshitz torque/γ0 (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

LoadFile(string) Slice

Load a data file (ovf or dump)

methods: CPUAccess( ) Comp(int) DevPtr(int) Disable( ) Free( ) GPUAccess( ) Get(int int int) Host( ) HostCopy( ) Index(int int int) IsNil( ) Len( ) MemType( ) Scalars( ) Set(int int int float64) SetScalar(int int int float64) SetVector(int int int data.Vector) Size( ) Tensors( ) Vectors( )

examples: [5]

log(float64) float64

Log returns the natural logarithm of x.

examples: [2] [4]

log10(float64) float64

Log10 returns the decimal logarithm of x. The special cases are the same as for Log.

log1p(float64) float64

Log1p returns the natural logarithm of 1 plus its argument x. It is more accurate than Log(1 + x) when x is near zero.

log2(float64) float64

Log2 returns the binary logarithm of x. The special cases are the same as for Log.

logb(float64) float64

Logb returns the binary exponent of x.

examples: [2]

m

Reduced magnetization (unit length)

methods: Average( ) Buffer( ) Comp(int) EvalTo(Slice) GetCell(int int int) LoadFile(string) Quantity( ) Region(int) Set(Config) SetArray(Slice) SetCell(int int int data.Vector)

SetInShape(Shape Config) SetRegion(int Config)

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

m_full

Unnormalized magnetization (A/m)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

Madd(Quantity, Quantity, float64, float64) *mAddition

Weighted addition: Madd(Q1,Q2,c1,c2) = c1*Q1 + c2*Q2

methods: EvalTo(Slice)

Masked(Quantity, Shape) Quantity

Mask quantity with shape

methods: EvalTo( )

max(float64, float64) float64

Max returns the larger of x or y.

examples: [3] [12]

MaxAngle

maximum angle between neighboring spins (rad)

methods: Average( ) EvalTo(Slice) Get( )

MaxDt

Maximum time step the solver can take (s)

MaxErr

Maximum error per step the solver can tolerate (default = 1e-5)

maxTorque

Maximum torque/γ0, over all cells (T)

methods: Average( ) EvalTo(Slice) Get( )

MFM

MFM image (arb.)

methods: Average( ) EvalTo(Slice) Region(int)

examples: [9]

MFMDipole

Height of vertically magnetized part of MFM tip

MFMLift

MFM lift height

examples: [9]

min(float64, float64) float64

Min returns the smaller of x or y.

examples: [3] [6]

MinDt

Minimum time step the solver can take (s)

Minimize()

Use steepest conjugate gradient method to minimize the total energy

examples: [3] [6]

MinimizerSamples

Number of max dM to collect for Minimize convergence check.

MinimizerStop

Stopping max dM for Minimize

examples: [3]

mod(float64, float64) float64

Mod returns the floating-point remainder of x/y. The magnitude of the result is less than y and its sign agrees with that of x.

Msat

Saturation magnetization (A/m)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [1] [2] [3] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

Mu0

Vacuum permeability (Tm/A)

examples: [2]

Mul(Quantity, Quantity) Quantity

Point-wise product of two quantities

methods: EvalTo( )

examples: [10] [11]

MulMV(Quantity, Quantity, Quantity, Quantity) Quantity

Matrix-Vector product: MulMV(AX, AY, AZ, m) = (AX·m, AY·m, AZ·m). The arguments Ax, Ay, Az and m are quantities with 3 componets.

methods: EvalTo( )

NeelSkyrmion(int, int) Config

Néél skyrmion magnetization with given charge and core polarization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [5]

NEval

Total number of torque evaluations

methods: Average( ) EvalTo(Slice) Get( )

NewScalarMask(int, int, int) Slice

Makes a 3D array of scalars

methods: CPUAccess( ) Comp(int) DevPtr(int) Disable( ) Free( ) GPUAccess( ) Get(int int int) Host( ) HostCopy( ) Index(int int int) IsNil( ) Len( ) MemType( ) Scalars( ) Set(int int int int float64) SetScalar(int int int float64) SetVector(int int int data.Vector) Size( ) Tensors( ) Vectors( )

NewSlice(int, int, int, int) Slice

Makes a 4D array with a specified number of components (first argument) and a specified size nx,ny,nz (remaining arguments)

methods: CPUAccess( ) Comp(int) DevPtr(int) Disable( ) Free( ) GPUAccess( ) Get(int int int int) Host( ) HostCopy( ) Index(int int int) IsNil( ) Len( ) MemType( ) Scalars( ) Set(int int int int float64) SetScalar(int int int float64) SetVector(int int int data.Vector) Size( ) Tensors( ) Vectors( )

NewVectorMask(int, int, int) Slice

Makes a 3D array of vectors

methods: CPUAccess( ) Comp(int) DevPtr(int) Disable( ) Free( ) GPUAccess( ) Get(int int int) Host( ) HostCopy( ) Index(int int int) IsNil( ) Len( ) MemType( ) Scalars( ) Set(int int int int float64) SetScalar(int int int float64) SetVector(int int int data.Vector) Size( ) Tensors( ) Vectors( )

examples: [15]

NoDemagSpins

Disable magnetostatic interaction per region (default=0, set to 1 to disable). E.g.: NoDemagSpins.SetRegion(5, 1) disables the magnetostatic interaction in region 5.

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

norm(float64) float64

Standard normal distribution

examples: [5] [12]

Normalized(Quantity) Quantity

Normalize quantity

methods: EvalTo( )

examples: [12]

now() time.Time

Returns the current time

methods: Add(time.Duration) AddDate(int int int) After(time.Time) AppendFormat([]uint8 string) Before(time.Time) Clock( ) Date( ) Day( ) Equal(time.Time) Format(string) GobEncode( ) Hour( ) ISOWeek( ) In(*time.Location) IsZero( ) Local( ) Location( ) MarshalBinary( ) MarshalJSON( ) MarshalText( ) Minute( ) Month( ) Nanosecond( ) Round(time.Duration) Second( ) Sub(time.Time) Truncate(time.Duration) UTC( ) Unix( ) UnixNano( ) Weekday( ) Year( ) YearDay( ) Zone( )

OpenBC

Use open boundary conditions (default=false)

OutputFormat

Format for data files: OVF1_TEXT, OVF1_BINARY, OVF2_TEXT or OVF2_BINARY

OVF1_BINARY

OutputFormat = OVF1_BINARY sets binary OVF1 output

OVF1_TEXT

OutputFormat = OVF1_TEXT sets text OVF1 output

OVF2_BINARY

OutputFormat = OVF2_BINARY sets binary OVF2 output

OVF2_TEXT

OutputFormat = OVF2_TEXT sets text OVF2 output

PeakErr

Overall maxium error per step

methods: Average( ) EvalTo(Slice) Get( )

pi

examples: [4] [5] [6] [11] [13] [14] [15]

Pol

Electrical current polarization

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [5] [10] [11] [14]

pow(float64, float64) float64

Pow returns x**y, the base-x exponential of y.

examples: [2] [15]

pow10(int) float64

Pow10 returns 10**n, the base-10 exponential of n.

Print(...interface {})

Print to standard output

examples: [2]

rand() float64

Random number between 0 and 1

examples: [3] [5] [12] [15]

randExp() float64

Exponentially distributed random number between 0 and +inf, mean=1

randInt(int) int

Random non-negative integer

randNorm() float64

Standard normal random number

examples: [12]

RandomMag() Config

Random magnetization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [3] [5]

RandomMagSeed(int) Config

Random magnetization with given seed

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

randSeed(int)

Sets the random number seed

Rect(float64, float64) Shape

2D rectangle with size in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4] [6] [9] [11] [12] [15]

regions

Outputs the region index for each cell

methods: Average( ) EvalTo(Slice) GetCell(int int int) Gpu( ) HostArray( ) HostList( ) LoadFile(string) SetCell(int int int int)

examples: [7] [12]

Relax()

Try to minimize the total energy

examples: [1] [2] [3] [9] [10] [11]

RelaxTorqueThreshold

MaxTorque threshold for relax(). If set to -1 (default), relax() will stop when the average torque is steady or increasing.

remainder(float64, float64) float64

Remainder returns the IEEE 754 floating-point remainder of x/y.

RemoveCustomFields()

Removes all custom fields again

Run(float64)

Run the simulation for a time in seconds

examples: [1] [7] [10] [11] [12] [14] [15]

RunWhile(func() bool)

Run while condition function is true

Save(Quantity)

Save space-dependent quantity once, with auto filename

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

SaveAs(Quantity, string)

Save space-dependent quantity with custom filename

examples: [4] [5] [7] [9]

SetCellSize(float64, float64, float64)

Sets the X,Y,Z cell size in meters

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

SetGeom(Shape)

Sets the geometry to a given shape

examples: [4] [6] [7] [8] [9] [11] [12] [14]

SetGridSize(int, int, int)

Sets the number of cells for X,Y,Z

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

SetMesh(int, int, int, float64, float64, float64, int, int, int)

Sets GridSize, CellSize and PBC at the same time

SetPBC(int, int, int)

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.

SetSolver(int)

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

Shift(int)

Shifts the simulation by +1/-1 cells along X

examples: [15]

Shifted(Quantity, int, int, int) Quantity

Shifted quantity

methods: EvalTo( )

ShiftGeom

Whether Shift() acts on geometry

ShiftM

Whether Shift() acts on magnetization

examples: [15]

ShiftMagD

Upon shift, insert this magnetization from the bottom

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

ShiftMagL

Upon shift, insert this magnetization from the left

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

ShiftMagR

Upon shift, insert this magnetization from the right

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

examples: [15]

ShiftMagU

Upon shift, insert this magnetization from the top

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

ShiftRegions

Whether Shift() acts on regions

Sign(float64) float64

Signum function

sin(float64) float64

Sin returns the sine of the radian argument x.

examples: [5] [6] [13] [14] [15]

sinc(float64) float64

Sinc returns sin(x)/x. If x=0, then Sinc(x) returns 1.

since(time.Time) time.Duration

Returns the time elapsed since argument

methods: Hours( ) Microseconds( ) Milliseconds( ) Minutes( ) Nanoseconds( ) Round(time.Duration) Seconds( ) Truncate(time.Duration)

sinh(float64) float64

Sinh returns the hyperbolic sine of x.

Snapshot(Quantity)

Save image of quantity

SnapshotAs(Quantity, string)

Save image of quantity with custom filename

SnapshotFormat

Image format for snapshots: jpg, png or gif.

spinAngle

Angle between neighboring spins (rad)

methods: Average( ) EvalTo(Slice) Region(int)

sprint(...interface {}) string

Print all arguments to string with automatic formatting

sprintf(string, ...interface {}) string

Print to string with C-style formatting.

sqrt(float64) float64

Sqrt returns the square root of x.

examples: [2]

Square(float64) Shape

2D square with size in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [6]

step

Total number of time steps taken

examples: [3] [10]

Steps(int)

Run the simulation for a number of time steps

STTorque

Spin-transfer torque/γ0 (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

t

Total simulated time (s)

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

TableAdd(Quantity)

Add quantity as a column to the data table.

examples: [3] [11] [13]

TableAddVar(ScalarFunction, string, string)

Add user-defined variable + name + unit to data table.

TableAutoSave(float64)

Auto-save the data table every period (s). Zero disables save.

examples: [1] [11] [14]

TablePrint(...interface {})

Print anyting in the data table

TableSave()

Save the data table right now (appends one line).

examples: [3] [13]

tan(float64) float64

Tan returns the tangent of the radian argument x.

tanh(float64) float64

Tanh returns the hyperbolic tangent of x.

Temp

Temperature (K)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

ThermSeed(int)

Set a random seed for thermal noise

torque

Total torque/γ0 (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

TotalShift

Amount by which the simulation has been shifted (m).

true

trunc(float64) float64

Trunc returns the integer value of x.

TwoDomain(float64, float64, float64, float64, float64, float64, float64, float64, float64) Config

Twodomain magnetization with with given magnetization in left domain, wall, and right domain

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [5] [10] [11]

Uniform(float64, float64, float64) Config

Uniform magnetization in given direction

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [1] [2] [5] [6] [7] [13] [14] [15]

Universe() Shape

Entire space

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

Vector(float64, float64, float64) data.Vector

Constructs a vector with given components

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

examples: [1] [3] [5] [7] [10] [11] [12] [14] [15]

Vortex(int, int) Config

Vortex magnetization with given circulation and core polarization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [5] [8] [9] [12]

VortexWall(float64, float64, int, int) Config

Vortex wall magnetization with given mx in left and right domain and core circulation and polarization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [5]

xi

Non-adiabaticity of spin-transfer-torque

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [10] [11] [12]

XRange(float64, float64) Shape

Part of space between x1 (inclusive) and x2 (exclusive), in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4] [7]

y0(float64) float64

Y0 returns the order-zero Bessel function of the second kind.

y1(float64) float64

Y1 returns the order-one Bessel function of the second kind.

yn(int, float64) float64

Yn returns the order-n Bessel function of the second kind.

YRange(float64, float64) Shape

Part of space between y1 (inclusive) and y2 (exclusive), in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

ZRange(float64, float64) Shape

Part of space between z1 (inclusive) and z2 (exclusive), in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

mumax^3

GPU-accelerated micromagnetism

Syntax

The mumax^3 input syntax is a subset of Go's syntax, somewhat similar to C. It is case-independent however, so msat is the same as Msat or MSAT.

Defining variables

New variables are declared using `:=`. Variables have a fixed type, inferred from the declaration's right-hand-side. Assigning to existing variables is done using `=`. E.g.:

```
i := 7          // defines a new variable i, type automatically detected to be int
print(i)        // now we can use i
i = 5           // assign new value, don't use ':=' (attempt to re-declare)

str := "hello" // defines str, type automatically is string
//str = 1      // would fail, cannot assign int to string
```

Arithmetic

Most common arithmetic operations are possible. Also Go's math library and some common constants are available. For raise-to-the-power, pow(x,y) should be used.

```
x := pi*(3+4)/5
x = pow(x, 3)
x++
y := abs(cbrt(cosh(erf(erfc(gamma(J0(Y0(2))))))))
```

Control structures

Loops are possible as well:

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

Implicit functions

Some of the API features accept a function as argument (e.g.: `RunWhile(func()bool)`, or all input parameters). In that case, and *only* in this case, the argument is implicitly converted to a function, which is re-evaluated each time it's needed. E.g.:

```
value := sin(pi*t)  // value is a float64, RHS evaluated only once
Msat = value        // time-independent Msat
```

versus:

```
Msat = sin(pi*t)    // RHS converted to function, re-evaluted every time
```

Methods

Some of the API instances have methods defined on them. You can call methods on an instance by using '.' as in most object oriented programming languages. E.g.: a material parameter such as `Msat` has the method `SetRegion(int, float)` to set the value of the material parameter in a certain region:

```
Msat.SetRegion(1, 800e3) // Set Msat=520e3 in region 1
```

mumax^3

GPU-accelerated micromagnetism

Mesh size and geometry

The simulation mesh defines the size of the box around your magnet. It should be set at the beginning of the script. The number of cells should preferably be powers of two, or at least have small prime factors (2,3,5,7). E.g.:

```
Nx := 128
Ny := 64
Nz := 2
sizeX := 500e-9
sizeY := 250e-9
sizeZ := 10e-9
SetGridSize(Nx, Ny, Nz)
SetCellSize(sizeX/Nx, sizeY/Ny, sizeZ/Nz)
```

Periodic boundary conditions

Optionally, periodic boundary conditions can be enabled:

```
SetPBC(5, 0, 0)        // 5 extra images on left and right sides.
SetGridSize(128, 64, 1)
SetCellSize(5e-9, 5e-9, 5e-9)
```

Setting a nonzero PBC value in a direction enables wrap-around in that direction. The precise value passed determines how many repetitions are seen by the demag field. E.g., in the above example the demag field behaves as if 5 repetitions are present to the left and to the right side. Choosing a large number may cause long initialization time.

Resizing the mesh

The mesh can be changed at any later time in the simulation. This will cause the magnetization to be stretched onto the new mesh if needed, and the geometry and regions to be re-calculated. After resize some cells which had zero magnetization may now fall inside the magnet geometry, they will be initialized to random magnetization.

Setting the geometry

Optionally a magnet Shape other than the full simulation box can be specified. In order to set the geometry, you first need to define a shape.

```
 geometryShape := cylinder(400e-9, 20e-9).RotX(45*pi/180).Transl(1e-6,0,0)
SetGeom(geometryShape)
```

SetCellSize(float64, float64, float64)

Sets the X,Y,Z cell size in meters

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

SetGeom(Shape)

Sets the geometry to a given shape

examples: [4] [6] [7] [8] [9] [11] [12] [14]

SetGridSize(int, int, int)

Sets the number of cells for X,Y,Z

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

SetMesh(int, int, int, float64, float64, float64, int, int, int)

Sets GridSize, CellSize and PBC at the same time

SetPBC(int, int, int)

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.

EdgeSmooth

Geometry edge smoothing with edgeSmooth^3 samples per cell, 0=staircase, ~8=very smooth

examples: [4]

mumax^3

GPU-accelerated micromagnetism

Shapes

A shape is an abstract object which outlines an area in a 3D universe. Shapes are useful for different tasks, e.g.: to define the geometry of a magnet, to define material regions, or to set locally a specific initial magnetization configuration. One can specify primitive shapes, constructed at the origin (box center), and translate/rotate them if needed. All positions are specified in meters and the origin lies in the center of the simulation box. E.g.:

```
myShape := cylinder(400e-9, 20e-9).RotX(45*pi/180).Transl(1e-6,0,0))
anotherShape := Circle(400e-9).sub(Circle(200e-9))
```

Cell(int, int, int) Shape

Single cell with given integer index (i, j, k)

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

Circle(float64) Shape

2D Circle with diameter in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(

float64 float64 float64) Xor(Shape)

examples: [4] [6] [7] [8] [12]

Cone(float64, float64) Shape

3D Cone with diameter and height in meter. The top of the cone points in the +z direction.

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

Cuboid(float64, float64, float64) Shape

Cuboid with sides in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4]

Cylinder(float64, float64) Shape

3D Cylinder with diameter and height in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4] [5] [6]

Ellipse(float64, float64) Shape

2D Ellipse with axes in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [14]

Ellipsoid(float64, float64, float64) Shape

3D Ellipsoid with axes in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4]

GrainRoughness(float64, float64, float64, int) Shape

Grainy surface with different heights per grain with a typical grain size (first argument), minimal height (second argument), and maximal height (third argument). The last argument is a seed for the random number generator.

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

ImageShape(string) Shape

Use black/white image as shape

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4]

Layer(int) Shape

Single layer (along z), by integer index starting from 0

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4] [13] [14]

Layers(int, int) Shape

Part of space between cell layer1 (inclusive) and layer2 (exclusive), in integer indices

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4]

Rect(float64, float64) Shape

2D rectangle with size in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4] [6] [9] [11] [12] [15]

Square(float64) Shape

2D square with size in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [6]

Universe() Shape

Entire space

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

XRange(float64, float64) Shape

Part of space between x1 (inclusive) and x2 (exclusive), in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

examples: [4] [7]

YRange(float64, float64) Shape

Part of space between y1 (inclusive) and y2 (exclusive), in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

ZRange(float64, float64) Shape

Part of space between z1 (inclusive) and z2 (exclusive), in meter

methods: Add(Shape) Intersect(Shape) Inverse( ) Repeat(float64 float64 float64) RotX(float64) RotY(float64) RotZ(float64) Scale(float64 float64 float64) Sub(Shape) Transl(float64 float64 float64) Xor(Shape)

mumax^3

GPU-accelerated micromagnetism

Material regions

Optionally, up to 256 material regions can be defined. Since each cell is made from one material, it is associated with exactly one region. So *regions can not overlap*. Each cell is assigned material region 0 by default. It's a good idea to output `regions` to verify whether each cell is assigned to the intended region. Each region can have its own material parameters, and we can output averages over each region. E.g.:

```
DefRegion(1, circle(1e-6))
DefRegion(0, circle(1e-6).Inverse()) // redundant
save(regions)
Msat.SetRegion(1, 800e6)
tableAdd(m.Region(1))    // add average m over region 1 to table
```

DefRegion(int, Shape)

Define a material region with given index (0-255) and shape

examples: [7] [12] [13]

DefRegionCell(int, int, int, int)

Set a material region (first argument) in one cell by the index of the cell (last three arguments)

regions

Outputs the region index for each cell

Misc

Go to mumax3.9c API

methods: Average( ) EvalTo(Slice) GetCell(int int int) Gpu( ) HostArray( ) HostList( ) LoadFile(string) SetCell(int int int int)

examples: [7] [12]

mumax^3

GPU-accelerated micromagnetism

Initial magnetization

The initial magnetization is set by assigning a `Config` to m, setting it in separate regions, or by loading a file directly.

```
m = uniform(1, 0, 0)
m.SetRegion(1, vortex(1, 1))
m.LoadFile("config.ovf")
m.SetInShape(circle(50e-9), uniform(0,0,1))
```

m

Reduced magnetization (unit length)

methods: Average( ) Buffer( ) Comp(int) EvalTo(Slice) GetCell(int int int) LoadFile(string) Quantity( ) Region(int) Set(Config) SetArray(Slice) SetCell(int int int data.Vector) SetInShape(Shape Config) SetRegion(int Config)

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

Antivortex(int, int) Config

Antivortex magnetization with given circulation and core polarization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

BlochSkyrmion(int, int) Config

Bloch skyrmion magnetization with given chirality and core polarization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [5]

Conical(data.Vector, data.Vector, float64) Config

Conical state for given wave vector, cone direction, and cone angle

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

Helical(data.Vector) Config

Helical state for given wave vector

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

NeelSkyrmion(int, int) Config

Néél skyrmion magnetization with given charge and core polarization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [5]

RandomMag() Config

Random magnetization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [3] [5]

RandomMagSeed(int) Config

Random magnetization with given seed

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

TwoDomain(float64, float64, float64, float64, float64, float64, float64, float64, float64) Config

Twodomain magnetization with with given magnetization in left domain, wall, and right domain

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [5] [10] [11]

Uniform(float64, float64, float64) Config

Uniform magnetization in given direction

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [1] [2] [5] [6] [7] [13] [14] [15]

Vortex(int, int) Config

Vortex magnetization with given circulation and core polarization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [5] [8] [9] [12]

`VortexWall(float64, float64, int, int) Config`

Vortex wall magnetization with given mx in left and right domain and core circulation and polarization

methods: Add(float64 Config) RotZ(float64) Scale(float64 float64 float64) Transl(float64 float64 float64)

examples: [5]

mumax^3

GPU-accelerated micromagnetism

Material parameters

Assigning to a material parameter sets a value in all regions. E.g.:

```
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. E.g.:

```
f := 500e6
Ku1 = 500 * sin(2*pi*f*t)
```

Aex

Exchange stiffness (J/m)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [1] [2] [3] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

Misc

Go to mumax3.9c API

alpha

Landau-Lifshitz damping constant

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [1] [6] [7] [8] [10] [11] [12] [14] [15]

anisC1

Cubic anisotropy direction #1

methods: Average( ) Comp(int) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [12]

anisC2

Cubic anisotorpy directon #2

methods: Average( ) Comp(int) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [12]

anisU

Uniaxial anisotropy direction

methods: Average( ) Comp(int) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [7] [10] [15]

B1

First magneto-elastic coupling constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

B2

Second magneto-elastic coupling constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

Dbulk

Bulk Dzyaloshinskii-Moriya strength (J/m2)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

Dind

Interfacial Dzyaloshinskii-Moriya strength (J/m2)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

EpsilonPrime

Slonczewski secondairy STT term ε'

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [14]

FreeLayerThickness

Slonczewski free layer thickness (if set to zero (default), then the thickness will be deduced from the mesh size) (m)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

frozenspins

Defines spins that should be fixed

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

Kc1

1st order cubic anisotropy constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [12]

Kc2

2nd order cubic anisotropy constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

Kc3

3rd order cubic anisotropy constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

Ku1

1st order uniaxial anisotropy constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [7] [10] [15]

Ku2

2nd order uniaxial anisotropy constant (J/m3)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

Lambda

Slonczewski Λ parameter

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [14]

Msat

Saturation magnetization (A/m)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [1] [2] [3] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

NoDemagSpins

Disable magnetostatic interaction per region (default=0, set to 1 to disable). E.g.: NoDemagSpins.SetRegion(5, 1) disables the magnetostatic interaction in region 5.

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

Pol

Electrical current polarization

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [5] [10] [11] [14]

Temp

Temperature (K)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

xi

Non-adiabaticity of spin-transfer-torque

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [10] [11] [12]

mumax^3

GPU-accelerated micromagnetism

Excitation

Field or current excitations can be set in the same way as material parameters:

```
B_ext = vector(0.01, 1e-6*sin(2*pi*f*t), 0)
B_ext.SetRegion(1, vector(0, 0, 0.1))
```

Additionally, an arbitrary number of time- and space-dependent vector fields of the form `g(x,y,z) * f(t)` may be added. (E.g., to simulate the field of an antenna or an arbitrary current running through the magnet)

```
B_ext.Add(LoadFile("antenna.ovf"), sin(2*pi*f*t))
J.Add(LoadFile("current.ovf"), 1)
```

B_ext

Externally applied field (T)

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(data.Vector) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [1] [3] [15]

FixedLayer

Misc

Go to mumax3.9c API

Slonczewski fixed layer polarization

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(data.Vector) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [14]

J

Electrical current density (A/m2)

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(data.Vector) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [10] [11] [12] [13] [14] [15]

mumax^3

GPU-accelerated micromagnetism

Output quantities

The quantities listed below can be output. Also, derived quantities can be produced: the quantity restricted to a certain region or a single component. E.g.:

```
m           // magnetization quantity
m.Comp(0)   // x-component
m.Region(1) // magnetization in region 1 (0 elsewhere)
```

Misc

Go to mumax3.9c API

B_anis

Anisotropy field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_custom

User-defined field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_demag

Magnetostatic field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_eff

Effective field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_exch

Exchange field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_mel

Magneto-elastic filed (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

B_therm

Thermal field (T)

methods: AddTo(Slice) EvalTo(Slice)

DindCoupling

Average DMI coupling with neighbors (arb.)

methods: Average( ) EvalTo(Slice) Region(int)

dt

Time Step (s)

methods: Average( ) EvalTo(Slice) Get( )

E_anis

total anisotropy energy (J)

methods: Average( ) EvalTo(Slice) Get( )

E_custom

total energy of user-defined field (J)

methods: Average( ) EvalTo(Slice) Get( )

E_demag

Magnetostatic energy (J)

methods: Average( ) EvalTo(Slice) Get( )

E_exch

Total exchange energy (including the DMI energy) (J)

methods: Average( ) EvalTo(Slice) Get( )

E_mel

Magneto-elastic energy (J)

methods: Average( ) EvalTo(Slice) Get( )

E_therm

Thermal energy (J)

methods: Average( ) EvalTo(Slice) Get( )

E_total

total energy (J)

methods: Average( ) EvalTo(Slice) Get( )

examples: [13]

E_Zeeman

Zeeman energy (J)

methods: Average( ) EvalTo(Slice) Get( )

Edens_anis

Anisotropy energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_custom

Energy density of user-defined field. (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_demag

Magnetostatic energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_exch

Total exchange energy density (including the DMI energy density) (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_mel

Magneto-elastic energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_therm

Thermal energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_total

Total energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

Edens_Zeeman

Zeeman energy density (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

ExchCoupling

Average exchange coupling with neighbors (arb.)

methods: Average( ) EvalTo(Slice) Region(int)

examples: [12]

F_mel

Magneto-elastic force density (N/m3)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

geom

Cell fill fraction (0..1)

methods: Average( ) EvalTo(Slice) Gpu( )

examples: [4] [6] [7] [8] [9] [11] [12] [14]

LastErr

Error of last step

methods: Average( ) EvalTo(Slice) Get( )

LLtorque

Landau-Lifshitz torque/γ0 (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

m_full

Unnormalized magnetization (A/m)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

MaxAngle

maximum angle between neighboring spins (rad)

methods: Average( ) EvalTo(Slice) Get( )

maxTorque

Maximum torque/γ0, over all cells (T)

methods: Average( ) EvalTo(Slice) Get( )

MFM

MFM image (arb.)

methods: Average( ) EvalTo(Slice) Region(int)

examples: [9]

NEval

Total number of torque evaluations

methods: Average( ) EvalTo(Slice) Get( )

PeakErr

Overall maxium error per step

methods: Average( ) EvalTo(Slice) Get( )

spinAngle

Angle between neighboring spins (rad)

methods: Average( ) EvalTo(Slice) Region(int)

STTorque

Spin-transfer torque/γ0 (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

torque

Total torque/γ0 (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

mumax^3

GPU-accelerated micromagnetism

Scheduling output

All input and output quantities (as described above) can be saved in a space-dependent way (".ovf" file), or as spatial averages (table output). The data table ("table.txt") contains by default the time and average magnetization. More columns can be added with `TableAdd()`.

```
save(B_ext)

tableadd(B_ext)
tablesave()
```

Optionally, the output/averaging can be done over a single region:

```
save(m.Region(1))
TableAdd(m.Region(1))
```

User-defined variables can be added to the table with `TableAddVar()`.

```
myField := 0.42
TableAddVar(myField, "B_extra", "T")
myField = ...
```

AutoSave(Quantity, float64)

Auto save space-dependent quantity every period (s).

examples: [1] [10] [11] [14] [15]

AutoSnapshot(Quantity, float64)

Auto save image of quantity every period (s).

DUMP

OutputFormat = DUMP sets text DUMP output

FilenameFormat

printf formatting string for output filenames.

Flush()

Flush all pending output to disk.

Fprintln(string, ...interface {})

Print to file

OutputFormat

Format for data files: OVF1_TEXT, OVF1_BINARY, OVF2_TEXT or OVF2_BINARY

OVF1_BINARY

OutputFormat = OVF1_BINARY sets binary OVF1 output

OVF1_TEXT

OutputFormat = OVF1_TEXT sets text OVF1 output

OVF2_BINARY

OutputFormat = OVF2_BINARY sets binary OVF2 output

OVF2_TEXT

OutputFormat = OVF2_TEXT sets text OVF2 output

Print(...interface {})

Print to standard output

examples: [2]

Save(Quantity)

Save space-dependent quantity once, with auto filename

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

SaveAs(Quantity, string)

Save space-dependent quantity with custom filename

examples: [4] [5] [7] [9]

Snapshot(Quantity)

Save image of quantity

SnapshotAs(Quantity, string)

Save image of quantity with custom filename

SnapshotFormat

Image format for snapshots: jpg, png or gif.

sprint(...interface {}) string

Print all arguments to string with automatic formatting

sprintf(string, ...interface {}) string

Print to string with C-style formatting.

TableAdd(Quantity)

Add quantity as a column to the data table.

examples: [3] [11] [13]

TableAddVar(ScalarFunction, string, string)

Add user-defined variable + name + unit to data table.

TableAutoSave(float64)

Auto-save the data table every period (s). Zero disables save.

examples: [1] [11] [14]

TablePrint(...interface {})

Print anyting in the data table

TableSave()

Save the data table right now (appends one line).

examples: [3] [13]

mumax^3

GPU-accelerated micromagnetism

Running

`Run(time)` runs the simulation for a given time in seconds, using sensible error settings.

```
Run(1e-9)
```

More fine-grained control is provided by `RunWhile(`*`condition`*`)`, which runs as long as an arbitrary condition is met. E.g.:

```
mx := m.comp(0)
RunWhile(mx.average() < 0)   // search for switching field during reversal
```

Optionally, the solver accuracy may be fine-tuned. E.g.:

```
MaxDt = 1e-12
MinDt = 1e-15
MaxErr = 1e-6
```

Optionally, a different solver may be chosen (at any point) with `SetSolver(int)`. Currently available solver types:

- 6: RK56 (Fehlberg) solver. This is the highest order solver available, but which is typically not faster than the RK45 solver.
- 5: RK45 (Dormand-Prince) solver (the default). An accurate solver, very fast for magnetization dynamics at the cost of some memory usage.
- 4: Classical 4th-order Runge-Kutta method. Intended for simulations where a fixed, relatively large time step is desired.

- `3`: RK23 (Bogacki-Shampine) solver. A robust and reasonably fast solver with low memory requirements. Typically outperforms RK45 when relaxing the magnetization with little dynamics, so it used internally by `Relax()`.
- `2`: Adaptive Heun solver. Robust and uses very little memory but takes smaller time steps than the higher-order solvers. Also suited when a fixed, relatively small time step is desired.
- `1`: Euler solver (requires `FixDt = ...`, ignores other settings). Only useful in exceptional situations or for debugging.

E.g.:

```
SetSolver(2) // Heun
FixDt = 1e-15
```

Relax

`Relax()` tries to evolve the magnetization as closely as possible to the minimum energy state. This function assumes all excitations have been turned off (temperature, electrical current, time-dependent magnetic fields). During relax precession is disabled and the time `t` does not increase. There is no need to set high damping.

In general it is difficult to be sure the minimum energy state has been truly reached. Hence, relax may occasionally return after the energy has reached a local minimum, a saddle point, or a rather flat valley in the energy landscape.

Minimize

`Minimize()` is like Relax, but uses the conjugate gradient method to find the energy minimum. It is usually much faster than Relax, but is a bit less robust against divergence. E.g., a random starting configuration can be Relaxed, but may fail with Minimize. Minimize is very well suited for hysteresis calculations, where we are never far away from the ground state.

`Minimize()`

Use steepest conjugate gradient method to minimize the total energy

examples: [3] [6]

`Relax()`

Try to minimize the total energy

examples: [1] [2] [3] [9] [10] [11]

Run(float64)

Run the simulation for a time in seconds

examples: [1] [7] [10] [11] [12] [14] [15]

RunWhile(func() bool)

Run while condition function is true

Steps(int)

Run the simulation for a number of time steps

dt

Time Step (s)

methods: Average( ) EvalTo(Slice) Get( )

FixDt

Set a fixed time step, 0 disables fixed step (which is the default)

Headroom

Solver headroom (default = 0.8)

LastErr

Error of last step

methods: Average( ) EvalTo(Slice) Get( )

MaxDt

Maximum time step the solver can take (s)

MaxErr

Maximum error per step the solver can tolerate (default = 1e-5)

MinDt

Minimum time step the solver can take (s)

MinimizerSamples

Number of max dM to collect for Minimize convergence check.

MinimizerStop

Stopping max dM for Minimize

examples: [3]

NEval

Total number of torque evaluations

methods: Average( ) EvalTo(Slice) Get( )

PeakErr

Overall maxium error per step

methods: Average( ) EvalTo(Slice) Get( )

RelaxTorqueThreshold

MaxTorque threshold for relax(). If set to -1 (default), relax() will stop when the average torque is steady or increasing.

step

Total number of time steps taken

examples: [3] [10]

t

Total simulated time (s)

examples: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

`SetSolver(int)`

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

mumax^3

GPU-accelerated micromagnetism

Spin currents

The effect of spin-polarized currents on the magnetization dynamics can be modelled in different ways. In Mumax3 you can use the Zhang-Li model or the Slonczewski model. For both models, a spin-polarized current field needs to be defined. This is done by setting the current density field `J` and the polarization `Pol`.

Zhang-Li model

When using the the Zhang-Li model, it is possible to set the non-adiabaticity through the material parameter `xi`:

```
J = vector(1e12, 0, 0)
Pol = 1
xi = 0.1
```

Slonczewski model

To use the Slonczewski model, you need to define the magnetization configuration of the fixed layer. This fixed layer can be placed above or below the sample. The Slonczewski parameter and the prefactor of the secondary spin transfer torque term of the Slonczewski model can be set through the material parameters `Lambda` and `EpsilonPrime` respectively:

```
DisableZhangLiTorque = true
J = vector(1e12, 0, 0)
Pol = 0.6
FixedLayer = vector(1,0,0)
FixedLayerPosition = FIXEDLAYER_TOP
```

Misc

Go to mumax3.9c API

```
EpsilonPrime = 0.02
Lambda = 1
```

DisableSlonczewskiTorque

Disables Slonczewski torque (default=false)

DisableZhangLiTorque

Disables Zhang-Li torque (default=false)

EpsilonPrime

Slonczewski secondairy STT term ε'

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [14]

FixedLayer

Slonczewski fixed layer polarization

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(data.Vector) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [14]

FIXEDLAYER_BOTTOM

FixedLayerPosition = FIXEDLAYER_BOTTOM instructs mumax3 that fixed layer is underneath of the free layer

FIXEDLAYER_TOP

FixedLayerPosition = FIXEDLAYER_TOP instructs mumax3 that fixed layer is on top of the free layer

FixedLayerPosition

Position of the fixed layer: FIXEDLAYER_TOP, FIXEDLAYER_BOTTOM (default=FIXEDLAYER_TOP)

FreeLayerThickness

Slonczewski free layer thickness (if set to zero (default), then the thickness will be deduced from the mesh size) (m)

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

J

Electrical current density (A/m2)

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(data.Vector) SetRegion(int VectorFunction) SetRegionFn(int func() [3]float64)

examples: [10] [11] [12] [13] [14] [15]

Lambda

Slonczewski Λ parameter

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [14]

Pol

Electrical current polarization

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [5] [10] [11] [14]

xi

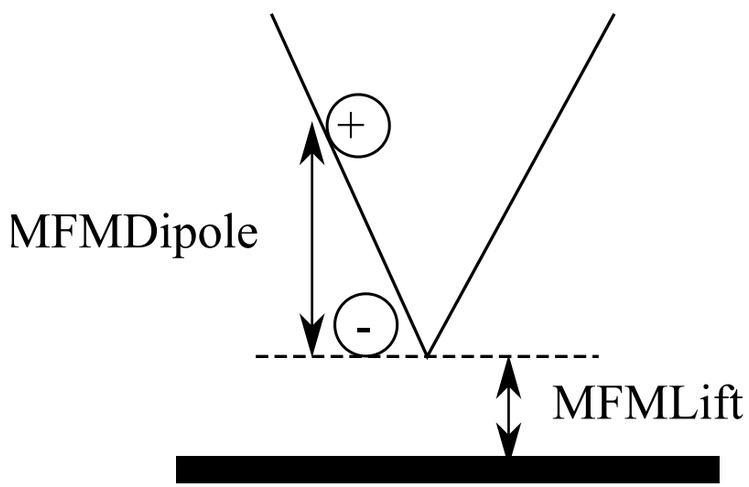
Non-adiabaticity of spin-transfer-torque

methods: Average( ) EvalTo(Slice) GetRegion(int) IsUniform( ) MSlice( ) Region(int) Set(float64) SetRegion(int ScalarFunction) SetRegionFuncGo(int func() float64) SetRegionValueGo(int float64)

examples: [10] [11] [12]

Magnetic Force Microscopy

Mumax3 has built-in generation of MFM images from a 2D magnetization. The MFM tip lift can be freely chosen. By default the tip magnetization is modeled as a point monopole at the apex. This is sufficient for most situations. Nevertheless, it is also possible to model partially magnetized tips by setting MFMDipole to the magnetized portion of the tip, in meters. E.g., if only the first 20nm of the tip is (vertically) magnetized, set MFMDipole=20e-9.



MFM

MFM image (arb.)

methods: Average( ) EvalTo(Slice) Region(int)

examples: [9]

MFMDipole

Height of vertically magnetized part of MFM tip

MFMLift

MFM lift height

examples: [9]

mumax^3

GPU-accelerated micromagnetism

Slicing and dicing output

To save storage space, it's possible to save only the part of the output we're interested in. This works on all output quantities (not only m)

```
save(m)                          // save full magnetization
save(m.Comp(0))                  // save only x-component
save(CropLayer(m, 13))           // save only layer 13
save(CropLayer(m.Comp(0), 13))   // save only x-component of layer 13
```

Or even:

```
mx   := m.Comp(0)
mx13 := CropLayer(mx, 13)
save(mx13)
tableAdd(mx13)
```

Crop(Quantity, int, int, int, int, int, int) *cropped

Crops a quantity to cell ranges [x1,x2[, [y1,y2[, [z1,z2[

methods: Average( ) EvalTo(Slice)

examples: [8]

CropLayer(Quantity, int) *cropped

Crops a quantity to a single layer

methods: Average( ) EvalTo(Slice)

CropRegion(Quantity, int) *cropped

Crops a quantity to a region

methods: Average( ) EvalTo(Slice)

CropX(Quantity, int, int) *cropped

Crops a quantity to cell ranges [x1,x2[

methods: Average( ) EvalTo(Slice)

CropY(Quantity, int, int) *cropped

Crops a quantity to cell ranges [y1,y2[

methods: Average( ) EvalTo(Slice)

examples: [8]

CropZ(Quantity, int, int) *cropped

Crops a quantity to cell ranges [z1,z2[

methods: Average( ) EvalTo(Slice)

mumax^3

GPU-accelerated micromagnetism

Moving simulation window

Mumax^3 can automatically shift the magnetization so that the simulation "window" stays centered on a region of interest. Shifting is done to keep a freely chosen magnetization component nearly zero. E.g.

```
ext_centerwall(0)
ext_rmSurfaceCharge(0, -1, 1)
TableAdd(TotalShift)
```

will try to keep `mx` (component 0, counting from 0) close to zero. If desired, one can override which "new" magnetization is inserted from the sides by setting `ShiftMagL` and `ShiftMagR`, though the default behaviour is usually OK.

Shift(int)

Shifts the simulation by +1/-1 cells along X

examples: [15]

ShiftGeom

Whether Shift() acts on geometry

ShiftM

Whether Shift() acts on magnetization

examples: [15]

ShiftMagD

Upon shift, insert this magnetization from the bottom

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

ShiftMagL

Upon shift, insert this magnetization from the left

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

ShiftMagR

Upon shift, insert this magnetization from the right

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

examples: [15]

ShiftMagU

Upon shift, insert this magnetization from the top

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

ShiftRegions

Whether Shift() acts on regions

TotalShift

Amount by which the simulation has been shifted (m).

mumax^3

GPU-accelerated micromagnetism

Extensions

Extensions are extra functionalities that are not officially supported. They are aimed at rather specific problems and may not work as expected for your particular situation. Their API and functionality may change in future releases.

ext_bubbledist

Bubble traveled distance (m)

methods: Average( ) EvalTo(Slice) Get( )

ext_BubbleMz

Center magnetization 1.0 or -1.0 (default = 1.0)

ext_bubblepos

Bubble core position (m)

methods: Average( ) EvalTo(Slice) Get( )

ext_bubblespeed

Bubble velocity (m/s)

methods: Average( ) EvalTo(Slice) Get( )

ext_centerBubble()

centerBubble shifts m after each step to keep the bubble position close to the center of the window

ext_centerWall(int)

centerWall(c) shifts m after each step to keep m_c close to zero

examples: [10] [11]

ext_corepos

Vortex core position (x,y) + polarization (z) (m)

methods: Average( ) EvalTo(Slice) Get( )

ext_dwpos

Position of the simulation window while following a domain wall (m)

methods: Average( ) EvalTo(Slice) Get( )

examples: [11]

ext_dwspeed

Speed of the simulation window while following a domain wall (m/s)

methods: Average( ) EvalTo(Slice) Get( )

ext_dwtilt

PMA domain wall tilt (rad)

methods: Average( ) EvalTo(Slice) Get( )

ext_dwxpos

Position of the simulation window while following a domain wall (m)

methods: Average( ) EvalTo(Slice) Get( )

ext_EnableUnsafe()

Deprecated. Only here to ensure maximal backwards compatibility with mumax3.9c.

ext_InterDind(int, int, float64)

Sets Dind coupling between two regions.

ext_InterExchange(int, int, float64)

Sets exchange coupling between two regions.

ext_make3dgrains(float64, int, int, Shape, int)

3D Voronoi tesselation over shape (grain size, starting region number, num regions, shape, seed)

ext_makegrains(float64, int, int)

Voronoi tesselation (grain size, num regions)

examples: [12] [15]

ext_phi

Azimuthal angle (rad)

methods: Average( ) EvalTo(Slice) Region(int)

ext_rmSurfaceCharge(int, float64, float64)

Compensate magnetic charges on the left and right sides of an in-plane magnetized wire.
Arguments: region, mx on left and right side, resp.

examples: [11]

ext_ScaleDind(int, int, float64)

Re-scales Dind coupling between two regions.

ext_ScaleExchange(int, int, float64)

Re-scales exchange coupling between two regions.

examples: [12] [13] [15]

ext_theta

Polar angle (rad)

methods: Average( ) EvalTo(Slice) Region(int)

ext_topologicalcharge

2D topological charge

methods: Average( ) EvalTo(Slice) Get( )

ext_topologicalchargedensity

2D topological charge density $m \cdot (\partial m/\partial x \times \partial m/\partial y)$ (1/m2)

methods: Average( ) EvalTo(Slice) Region(int)

ext_topologicalchargedensitylattice

2D topological charge density according to Berg and Lüscher (1/m2)

methods: Average( ) EvalTo(Slice) Region(int)

ext_topologicalchargelattice

2D topological charge according to Berg and Lüscher

methods: Average( ) EvalTo(Slice) Get( )

mumax^3

GPU-accelerated micromagnetism

Custom quantities

Using existing quantities, it is possible to define new custom quantities. E.g.: instead of using the pre-defined `ext_topologicalchargedensity` quantity, it is possible to define this quantity yourselves inside an input script:

```
cs := 1e-9
setcellsize(cs,cs,cs)
setgridsize(64,64,1)

// Use central finite differences to approximate the spatial derivatives of m
mL := Shifted(m,-1,0,0) // shift left
mR := Shifted(m,1,0,0)  // shift right
mD := Shifted(m,0,-1,0) // shift up
mU := Shifted(m,0,1,0)  // shift down
dmdx := Mul( Const(1/(2*cs)), Madd(mR,mL,1,-1) )
dmdy := Mul( Const(1/(2*cs)), Madd(mU,mD,1,-1) )

// Define the topological charge density
chargeDensity := Mul( Const(1/(4*pi)), Dot(m, Cross(dmdx,dmdy)))

// Save the topological charge density of a skyrmion
m = neelskyrmion(1,-1)
saveas(chargeDensity, "chargeDensity.ovf")
```

Add(Quantity, Quantity) Quantity

Add two quantities

methods: EvalTo( )

examples: [3] [4] [5] [11] [13] [15]

Const(float64) Quantity

Constant, uniform number

methods: EvalTo( )

ConstVector(float64, float64, float64) Quantity

Constant, uniform vector

methods: EvalTo( )

Cross(Quantity, Quantity) Quantity

Cross product of two vector quantities

methods: EvalTo( )

examples: [12]

Div(Quantity, Quantity) Quantity

Point-wise division of two quantities

methods: EvalTo( )

Dot(Quantity, Quantity) Quantity

Dot product of two vector quantities

methods: EvalTo( )

Madd(Quantity, Quantity, float64, float64) *mAddition

Weighted addition: Madd(Q1,Q2,c1,c2) = c1*Q1 + c2*Q2

methods: EvalTo(Slice)

Masked(Quantity, Shape) Quantity

Mask quantity with shape

methods: EvalTo( )

Mul(Quantity, Quantity) Quantity

Point-wise product of two quantities

methods: EvalTo( )

examples: [10] [11]

MulMV(Quantity, Quantity, Quantity, Quantity) Quantity

Matrix-Vector product: MulMV(AX, AY, AZ, m) = (AX·m, AY·m, AZ·m). The arguments Ax, Ay, Az and m are quantities with 3 componets.

methods: EvalTo( )

Shifted(Quantity, int, int, int) Quantity

Shifted quantity

methods: EvalTo( )

mumax^3

GPU-accelerated micromagnetism

Custom effective field terms

It is possible to define additional effective field terms by promoting a custom quantity to an effective field term. The corresponding energy density term can also be added by promoting a custom quantity. E.g.: instead of using the existing anistropy field in mumax3, you could define the uniaxial anisotropy field (and the corresponding energy density) yourselves:

```
Ms := 1100e3
K  := 0.5e6
u  := ConstVector(1, 0, 0)
anisField := Mul( Const(2*K/Ms)  , Mul( Dot(u, m), u))
anisEdens := Mul( Const(-0.5*Ms) , Dot( anisField, m))

AddFieldTerm(anisField) // promote anisField to an effective field term
AddEdensTerm(anisEdens) // promote anisEdens to an energy density term

tableAdd(E_custom)  // Add a column with the energy related to the custom field
```

AddEdensTerm(Quantity)

Add an expression to Edens.

AddFieldTerm(Quantity)

Add an expression to B_eff.

B_custom

User-defined field (T)

methods: Average( ) Comp(int) EvalTo(Slice) HostCopy( ) Region(int)

E_custom

total energy of user-defined field (J)

methods: Average( ) EvalTo(Slice) Get( )

Edens_custom

Energy density of user-defined field. (J/m3)

methods: Average( ) EvalTo(Slice) Region(int)

RemoveCustomFields()

Removes all custom fields again

mumax^3

GPU-accelerated micromagnetism

Misc

Other available functions.

abs(float64) float64

Abs returns the absolute value of x.

acos(float64) float64

Acos returns the arccosine, in radians, of x.

acosh(float64) float64

Acosh returns the inverse hyperbolic cosine of x.

asin(float64) float64

Asin returns the arcsine, in radians, of x.

asinh(float64) float64

Asinh returns the inverse hyperbolic sine of x.

atan(float64) float64

Atan returns the arctangent, in radians, of x.

atan2(float64, float64) float64

Atan2 returns the arc tangent of y/x, using the signs of the two to determine the quadrant of the return value.

atanh(float64) float64

Atanh returns the inverse hyperbolic tangent of x.

cbrt(float64) float64

Cbrt returns the cube root of x.

ceil(float64) float64

Ceil returns the least integer value greater than or equal to x.

cos(float64) float64

Cos returns the cosine of the radian argument x.

examples: [6] [13] [14]

cosh(float64) float64

Cosh returns the hyperbolic cosine of x.

DemagAccuracy

Controls accuracy of demag kernel

DoPrecess

Enables LL precession (default=true)

EnableDemag

Enables/disables demag (default=true)

erf(float64) float64

Erf returns the error function of x.

erfc(float64) float64

Erfc returns the complementary error function of x.

Exit()

Exit from the program

exp(float64) float64

Exp returns e**x, the base-e exponential of x.

examples: [15]

exp2(float64) float64

Exp2 returns 2**x, the base-2 exponential of x.

Expect(string, float64, float64, float64)

Used for automated tests: checks if a value is close enough to the expected value

ExpectV(string, data.Vector, data.Vector, float64)

Used for automated tests: checks if a vector is close enough to the expected value

expm1(float64) float64

Expm1 returns e**x - 1, the base-e exponential of x minus 1. It is more accurate than Exp(x) - 1 when x is near zero.

exx

exx component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

exy

exy component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

exz

exz component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

eyy

eyy component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

eyz

eyz component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

ezz

ezz component of the strain tensor

methods: Add(Slice ScalarFunction) AddGo(Slice func() float64) AddTo(Slice) Average( ) Comp(int) EvalTo(Slice) IsUniform( ) MSlice( ) Region(int) RemoveExtraTerms( ) Set(float64) SetRegion(int ScalarFunction) SetRegionFn(int func() [3]float64)

false

floor(float64) float64

Floor returns the greatest integer value less than or equal to x.

gamma(float64) float64

Gamma returns the Gamma function of x.

GammaLL

Gyromagnetic ratio in rad/Ts

heaviside(float64) float64

Returns 1 if x>0, 0 if x<0, and 0.5 if x==0

hypot(float64, float64) float64

Hypot returns Sqrt(p*p + q*q), taking care to avoid unnecessary overflow and underflow.

ilogb(float64) int

Ilogb returns the binary exponent of x as an integer.

examples: [2]

Index2Coord(int, int, int) data.Vector

Convert cell index to x,y,z coordinate in meter

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

examples: [15]

inf

Inf returns positive infinity if sign >= 0, negative infinity if sign < 0.

examples: [4] [7] [11]

isInf(float64, int) bool

IsInf reports whether f is an infinity, according to sign. If sign > 0, IsInf reports whether f is positive infinity. If sign < 0, IsInf reports

isNaN(float64) bool

IsNaN reports whether f is an IEEE 754 “not-a-number” value.

j0(float64) float64

J0 returns the order-zero Bessel function of the first kind.

j1(float64) float64

J1 returns the order-one Bessel function of the first kind.

jn(int, float64) float64

Jn returns the order-n Bessel function of the first kind.

ldexp(float64, int) float64

Ldexp is the inverse of Frexp. It returns frac × 2**exp.

LoadFile(string) Slice

Load a data file (ovf or dump)

methods: CPUAccess( ) Comp(int) DevPtr(int) Disable( ) Free( ) GPUAccess( ) Get(int int int int) Host( ) HostCopy( ) Index(int int int) IsNil( ) Len( ) MemType( ) Scalars( ) Set(int int int int float64) SetScalar(int int int float64) SetVector(int int int data.Vector) Size( ) Tensors( ) Vectors( )

examples: [5]

log(float64) float64

Log returns the natural logarithm of x.

examples: [2] [4]

log10(float64) float64

Log10 returns the decimal logarithm of x. The special cases are the same as for Log.

log1p(float64) float64

Log1p returns the natural logarithm of 1 plus its argument x. It is more accurate than Log(1 + x) when x is near zero.

log2(float64) float64

Log2 returns the binary logarithm of x. The special cases are the same as for Log.

logb(float64) float64

Logb returns the binary exponent of x.

examples: [2]

max(float64, float64) float64

Max returns the larger of x or y.

examples: [3] [12]

min(float64, float64) float64

Min returns the smaller of x or y.

examples: [3] [6]

mod(float64, float64) float64

Mod returns the floating-point remainder of x/y. The magnitude of the result is less than y and its sign agrees with that of x.

Mu0

Vacuum permeability (Tm/A)

examples: [2]

NewScalarMask(int, int, int) Slice

Makes a 3D array of scalars

methods: CPUAccess( ) Comp(int) DevPtr(int) Disable( ) Free( ) GPUAccess( ) Get(int int int) Host( ) HostCopy( ) Index(int int int) IsNil( ) Len( ) MemType( ) Scalars( ) Set(

int int int float64) SetScalar(int int int float64) SetVector(int int int data.Vector) Size( ) Tensors( ) Vectors( )

NewSlice(int, int, int, int) Slice

Makes a 4D array with a specified number of components (first argument) and a specified size nx,ny,nz (remaining arguments)

methods: CPUAccess( ) Comp(int) DevPtr(int) Disable( ) Free( ) GPUAccess( ) Get(int int int int) Host( ) HostCopy( ) Index(int int int) IsNil( ) Len( ) MemType( ) Scalars( ) Set(int int int int float64) SetScalar(int int int float64) SetVector(int int int data.Vector) Size( ) Tensors( ) Vectors( )

NewVectorMask(int, int, int) Slice

Makes a 3D array of vectors

methods: CPUAccess( ) Comp(int) DevPtr(int) Disable( ) Free( ) GPUAccess( ) Get(int int int int) Host( ) HostCopy( ) Index(int int int) IsNil( ) Len( ) MemType( ) Scalars( ) Set(int int int int float64) SetScalar(int int int float64) SetVector(int int int data.Vector) Size( ) Tensors( ) Vectors( )

examples: [15]

norm(float64) float64

Standard normal distribution

examples: [5] [12]

Normalized(Quantity) Quantity

Normalize quantity

methods: EvalTo( )

examples: [12]

now() time.Time

Returns the current time

methods: Add(time.Duration) AddDate(int int int) After(time.Time) AppendFormat([]uint8 string) Before(time.Time) Clock( ) Date( ) Day( ) Equal(time.Time) Format(string) GobEncode( ) Hour( ) ISOWeek( ) In(*time.Location) IsZero( ) Local( ) Location( ) MarshalBinary( ) MarshalJSON( ) MarshalText( ) Minute( ) Month( ) Nanosecond( ) Round(time.Duration) Second( ) Sub(time.Time) Truncate(time.Duration) UTC( ) Unix( ) UnixNano( ) Weekday( ) Year( ) YearDay( ) Zone( )

OpenBC

Use open boundary conditions (default=false)

pi

examples: [4] [5] [6] [11] [13] [14] [15]

pow(float64, float64) float64

Pow returns x**y, the base-x exponential of y.

examples: [2] [15]

pow10(int) float64

Pow10 returns 10**n, the base-10 exponential of n.

rand() float64

Random number between 0 and 1

examples: [3] [5] [12] [15]

randExp() float64

Exponentially distributed random number between 0 and +inf, mean=1

randInt(int) int

Random non-negative integer

randNorm() float64

Standard normal random number

examples: [12]

randSeed(int)

Sets the random number seed

remainder(float64, float64) float64

Remainder returns the IEEE 754 floating-point remainder of x/y.

Sign(float64) float64

Signum function

sin(float64) float64

Sin returns the sine of the radian argument x.

examples: [5] [6] [13] [14] [15]

sinc(float64) float64

Sinc returns sin(x)/x. If x=0, then Sinc(x) returns 1.

since(time.Time) time.Duration

Returns the time elapsed since argument

methods: Hours( ) Microseconds( ) Milliseconds( ) Minutes( ) Nanoseconds( ) Round(time.Duration) Seconds( ) Truncate(time.Duration)

sinh(float64) float64

Sinh returns the hyperbolic sine of x.

sqrt(float64) float64

Sqrt returns the square root of x.

examples: [2]

tan(float64) float64

Tan returns the tangent of the radian argument x.

tanh(float64) float64

Tanh returns the hyperbolic tangent of x.

ThermSeed(int)

Set a random seed for thermal noise

true

trunc(float64) float64

Trunc returns the integer value of x.

Vector(float64, float64, float64) data.Vector

Constructs a vector with given components

methods: Add(data.Vector) Cross(data.Vector) Div(float64) Dot(data.Vector) Len( ) MAdd(float64 data.Vector) Mul(float64) Sub(data.Vector) X( ) Y( ) Z( )

examples: [1] [3] [5] [7] [10] [11] [12] [14] [15]

y0(float64) float64

Y0 returns the order-zero Bessel function of the second kind.

y1(float64) float64

Y1 returns the order-one Bessel function of the second kind.

yn(int, float64) float64

Yn returns the order-n Bessel function of the second kind.