OOMMF
eXtensible
Solver

M. J. Donahue and D. G. Porter

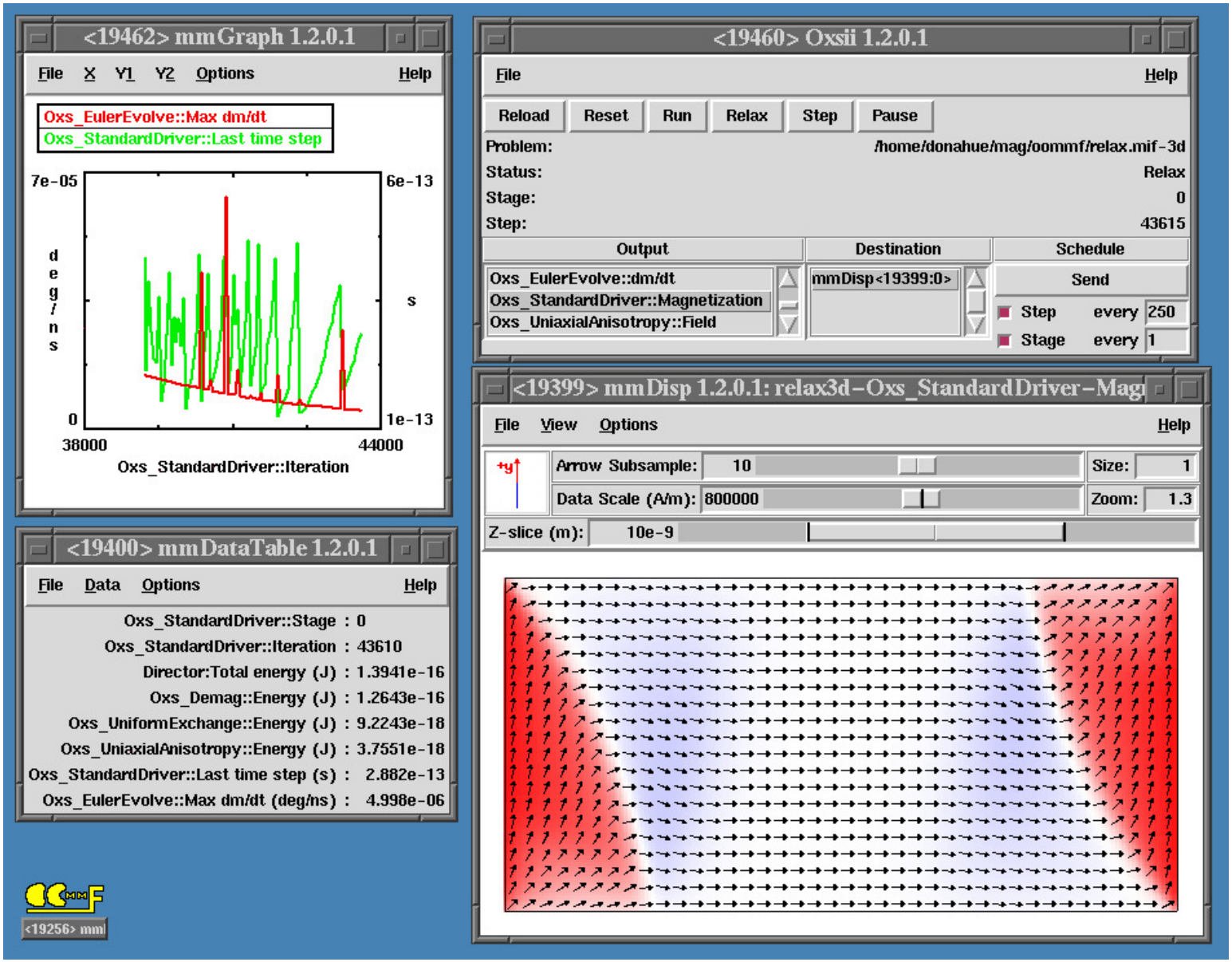
NIST, Gaithersburg, MD USA

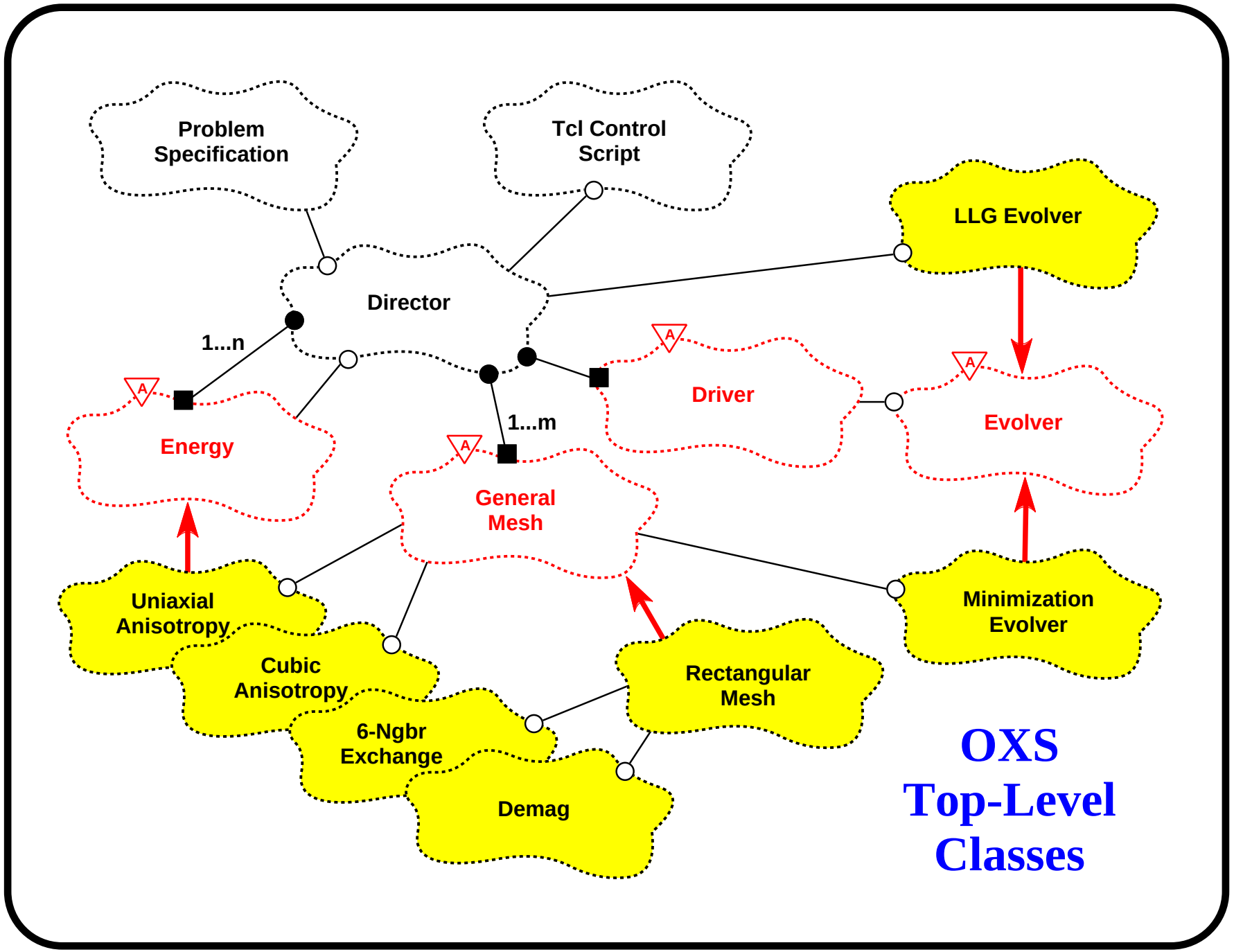
OOMMF: http://math.nist.gov/oommf

μMAG: http://www.ctcms.nist.gov/~rdm/mumag.org.html

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

1901-2001
NIST CENTENNIAL


<19462> mmGraph 1.2.0.1
File X Y1 Y2 Options Help
Oxs_EulerEvolve::Max dm/dt
Oxs_StandardDriver::Last time step
7e-05
6e-13
deg/ns
s
0
1e-13
38000
44000
Oxs_StandardDriver::Iteration
<19460> Oxsii 1.2.0.1
File Help
Reload Reset Run Relax Step Pause
Problem: /home/donahue/mag/oommf/relax.mif-3d
Status: Relax
Stage: 0
Step: 43615
Output
Destination
Schedule
Oxs_EulerEvolve::dm/dt
Oxs_StandardDriver::Magnetization
Oxs_UniaxialAnisotropy::Field
mmDisp<19399:0>
Send
Step every 250
Stage every 1
<19399> mmDisp 1.2.0.1: relax3d-Oxs_StandardDriver-Magn
File View Options Help
+y
Arrow Subsample: 10
Size: 1
Data Scale (A/m): 800000
Zoom: 1.3
Z-slice (m): 10e-9
<19400> mmDataTable 1.2.0.1
File Data Options Help
Oxs_StandardDriver::Stage : 0
Oxs_StandardDriver::Iteration : 43610
Director:Total energy (J) : 1.3941e-16
Oxs_Demag::Energy (J) : 1.2643e-16
Oxs_UniformExchange::Energy (J) : 9.2243e-18
Oxs_UniaxialAnisotropy::Energy (J) : 3.7551e-18
Oxs_StandardDriver::Last time step (s) : 2.882e-13
Oxs_EulerEvolve::Max dm/dt (deg/ns) : 4.998e-06
<19256> mmf


Problem Specification
Tcl Control Script
LLG Evolver
Director
1...n
1...m
A
Energy
Driver
Evolver
General Mesh
Uniaxial Anisotropy
Cubic Anisotropy
6-Ngbr Exchange
Demag
Rectangular Mesh
Minimization Evolver
OXS Top-Level Classes

Micromagnetic Equations

Landau-Lifshitz-Gilbert:

$$\frac{d\mathbf{M}}{dt} = \frac{-\omega}{1+\lambda^2}\mathbf{M}\times\mathbf{H}_{\text{eff}} - \frac{\lambda\,\omega}{(1+\lambda^2)M_{\text{s}}}\mathbf{M}\times(\mathbf{M}\times\mathbf{H}_{\text{eff}})$$

$$\mathbf{H}_{\text{eff}} = -\frac{1}{\mu_0}\frac{\partial E_{\text{density}}}{\partial \mathbf{M}}$$

Energies:

$$E_{\text{exchange}} = \frac{A}{M_s^2}\left(|\nabla M_x|^2 + |\nabla M_y|^2 + |\nabla M_z|^2\right)$$

$$E_{\text{anis,cubic}} = \frac{K_1}{M_s^4}(M_x^2M_y^2 + M_y^2M_z^2 + M_z^2M_x^2)$$

$$E_{\text{demag}} = \frac{\mu_0}{8\pi}\mathbf{M}(r)\cdot\left[\int_V \nabla\cdot\mathbf{M}(\mathbf{r}')\frac{\mathbf{r}-\mathbf{r}'}{|\mathbf{r}-\mathbf{r}'|^3}\,d^3r' - \int_S \hat{\mathbf{n}}\cdot\mathbf{M}(\mathbf{r}')\frac{\mathbf{r}-\mathbf{r}'}{|\mathbf{r}-\mathbf{r}'|^3}\,d^2r'\right]$$

$$E_{\text{Zeeman}} = -\mu_0\mathbf{M}\cdot\mathbf{H}_{\text{ext}}$$

Sample MIF 2.0 File

```
# MIF 2.0
Specify Oxs_SectionAtlas:atlas {
  world { Oxs_RectangularSection {
    xrange {0 600e-9} yrange {0 600e-9} zrange {0 40e-9}
  }}
}

Specify Oxs_RectangularMesh:mesh {
  cellsize {5e-9 5e-9 5e-9}
  atlas :atlas
}

- Specify Oxs_SimpleAnisotropy {
-   K1 5.2e5
-   axis {0 0 1}
- }

+ Specify My_ExtendedAnisotropy {
+   K1 5.2e5
+   K2 1.2e5
+   axis {0 0 1}
+ }

Specify Oxs_UniformExchange { A 30e-12 }

Specify Oxs_Demag {}
```

```
Specify Oxs_EulerEvolve {
  alpha 0.5
  start_dm 0.01
}

Specify Oxs_StandardDriver {
 evolver Oxs_EulerEvolve
 mesh :mesh
 min_timestep 1e-15
 max_timestep 10e-9
 stopping_dm_dt 0.01
 Ms { Oxs_UniformScalarFieldInit { value 1.4e6 }}
- m0 { Oxs_ScriptVectorFieldInit {
-   script  Box
-   norm  1
- }}
+ m0 { Oxs_FileVectorFieldInit { file "simpleanis-final.omf" }}
}

proc Box { x y z xmin ymin zmin xmax ymax zmax } {
    set tx [expr {double($x-$xmin)/double($xmax-$xmin)}]
    set ty [expr {double($y-$ymin)/double($ymax-$ymin)}]
    if { $tx<0.1 && $ty<0.9 } { return "0 -1 0" }
    if { $ty<0.1 } { return "1 0 0" }
    if { $tx>0.9 } { return "0 1 0" }
    if { $ty>0.9 } { return "-1 0 0" }
    return "0 0 1"
}
```

Adding a New Energy Term

1. Copy sample `.h` and `.cc` files to `oommf/app/oxs/local`.
2. Change names.
3. Add new code.
4. Run `pimake`.
5. Add new term to `MIF` input file.

NB: Modify no files in OOMMF distribution!

Uniaxial Anisotropy:

$$\text{Simple form: } E_{\text{anis}} = K_1 \sin^2 \phi$$

$$\text{Extended form: } E_{\text{anis}} = K_1 \sin^2 \phi + K_2 \sin^4 \phi$$

where ϕ is angle between $\mathbf{m}$ and $\mathbf{u}$.

Sample Anisotropy Header File

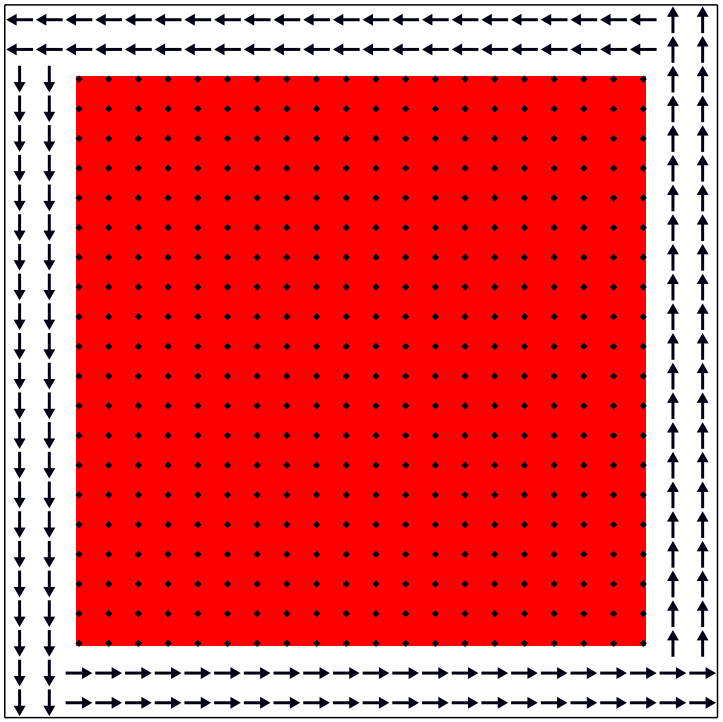
```
// Sample uniaxial anisotropy, derived from Oxs_Energy class.
#include "nb.h"
#include "threevector.h"
#include "energy.h"
#include "key.h"
#include "simstate.h"
#include "mesh.h"
#include "meshvalue.h"
/* End includes */

- class Oxs_SimpleAnisotropy:public Oxs_Energy {
+ class My_ExtendedAnisotropy:public Oxs_Energy {
private:
- REAL8m K1;
+ REAL8m K1,K2;
  ThreeVector axis;
public:
  virtual const char* ClassName() const; // ClassName() is
  /// automatically generated by the OXS_EXT_REGISTER macro.

- Oxs_SimpleAnisotropy(const char* name,  // Child instance id
+ My_ExtendedAnisotropy(const char* name, // Child instance id
       Oxs_Director* newdtr,    // App director
       Tcl_Interp* safe_interp, // Safe interpreter
       const char* argstr);     // MIF input block parameters

- virtual ~Oxs_SimpleAnisotropy() {}
+ virtual ~My_ExtendedAnisotropy() {}

  virtual void
  GetEnergyAndField(const Oxs_SimState& state,
                    Oxs_MeshValue<REAL8m>& energy,
                    Oxs_MeshValue<ThreeVector>& field
                    ) const;
};
```

Sample Anisotropy Source File

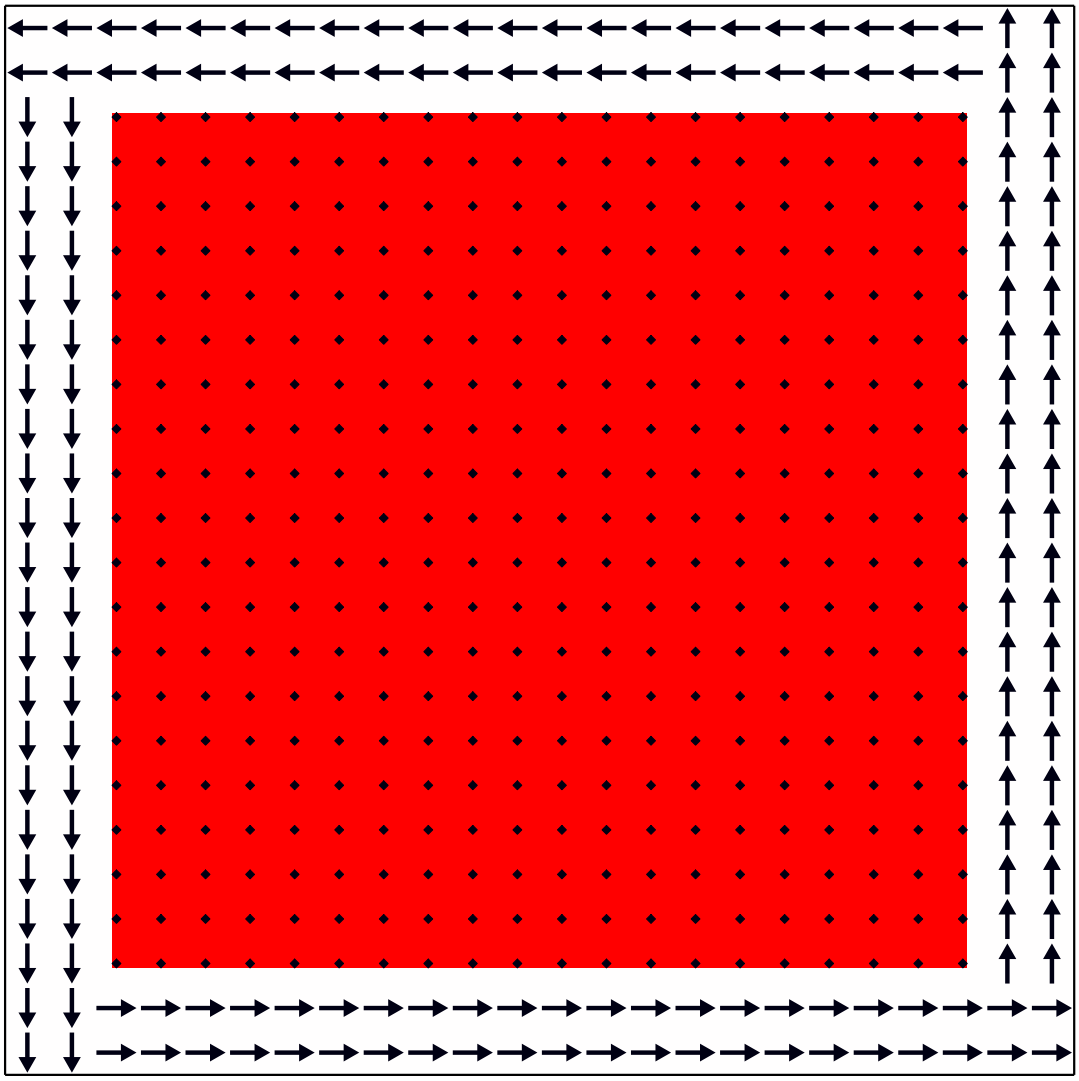
```
// Sample uniaxial anisotropy, derived from Oxs_Energy class.

- #include "simpleanisotropy.h"
+ #include "myanisotropy.h"

// Oxs_Ext registration support
- OXS_EXT_REGISTER(Oxs_SimpleAnisotropy);
+ OXS_EXT_REGISTER(My_ExtendedAnisotropy);

/* End includes */

// Constructor
- Oxs_SimpleAnisotropy::Oxs_SimpleAnisotropy(
+ My_ExtendedAnisotropy::My_ExtendedAnisotropy(
  const char* name,     // Child instance id
  Oxs_Director* newdtr, // App director
  Tcl_Interp* safe_interp, // Safe interpreter
  const char* argstr)   // MIF input block parameters
  : Oxs_Energy(name,newdtr,safe_interp,argstr)
{
  // Process arguments
  K1=GetRealInitValue("K1");
+ K2=GetRealInitValue("K2");
  axis=GetThreeVectorInitValue("axis");
  VerifyAllInitArgsUsed();
}
```

```
// Energy and field calculation code
- void Oxs_SimpleAnisotropy::GetEnergyAndField
+ void My_ExtendedAnisotropy::GetEnergyAndField
(const Oxs_SimState& state,
 Oxs_MeshValue<REAL8m>& energy,
 Oxs_MeshValue<ThreeVector>& field
 ) const
{
  const Oxs_MeshValue<REAL8m>& Ms_inverse=*(state.Ms_inverse);
  const Oxs_MeshValue<ThreeVector>& spin =state.spin;
  UINT4m size = state.mesh->Size();
  for(UINT4m i=0;i<size;++i) {
    if(Ms_inverse[i]==0.0) {
      energy[i]=0.0;
      field[i].Set(0.,0.,0.);
    } else {
      REAL8m dot = axis*spin[i];
      REAL8m dotsq = dot*dot;
-     energy[i] = -K1*dotsq;
-     REAL8m fieldmag = (2./MU0)*K1*dot*Ms_inverse[i];
+     energy[i] = ((dotsq-2)*K2-K1)*dotsq;
+     REAL8m fieldmag
+           = (-2./MU0)*((dotsq-1)*2*K2-K1)*dot*Ms_inverse[i];
      field[i]  = fieldmag * axis;
    }
  }
}
```

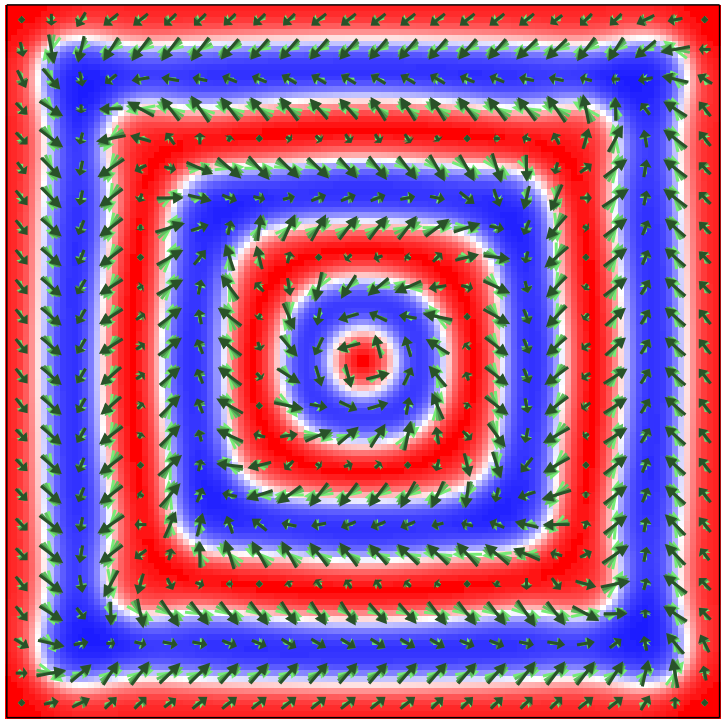
Initial Magnetization Configuration



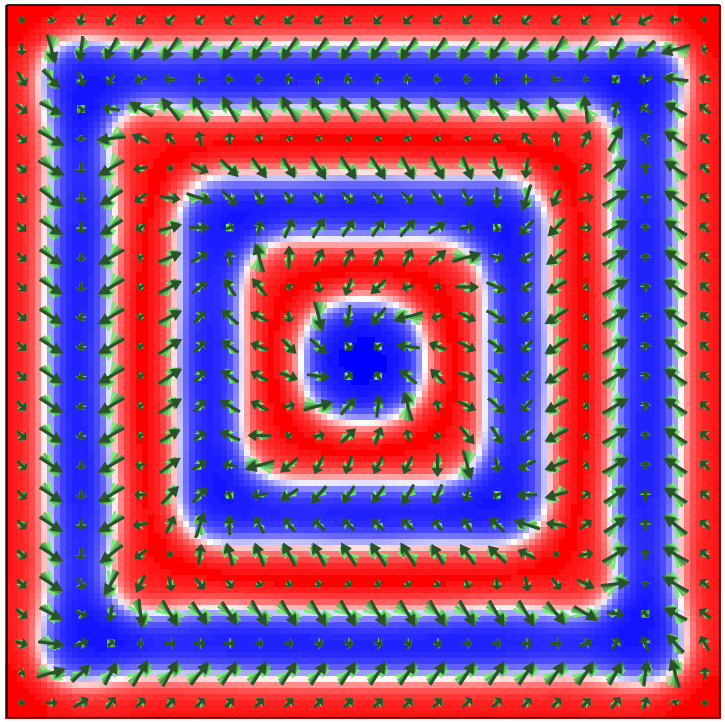
Initial Magnetization Configuration



Final Magnetization Configuration



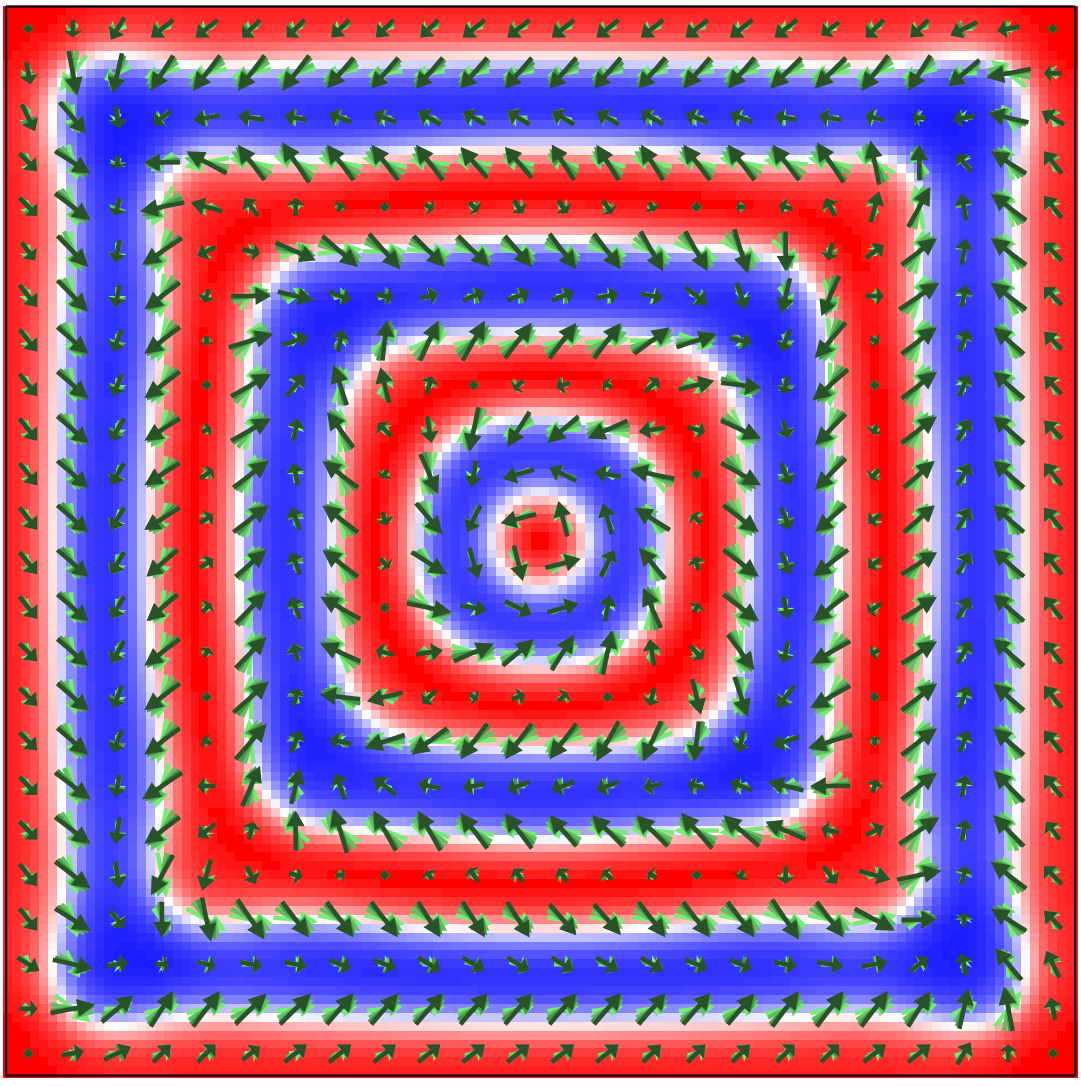
Simple Anisotropy



Extended Anistropy

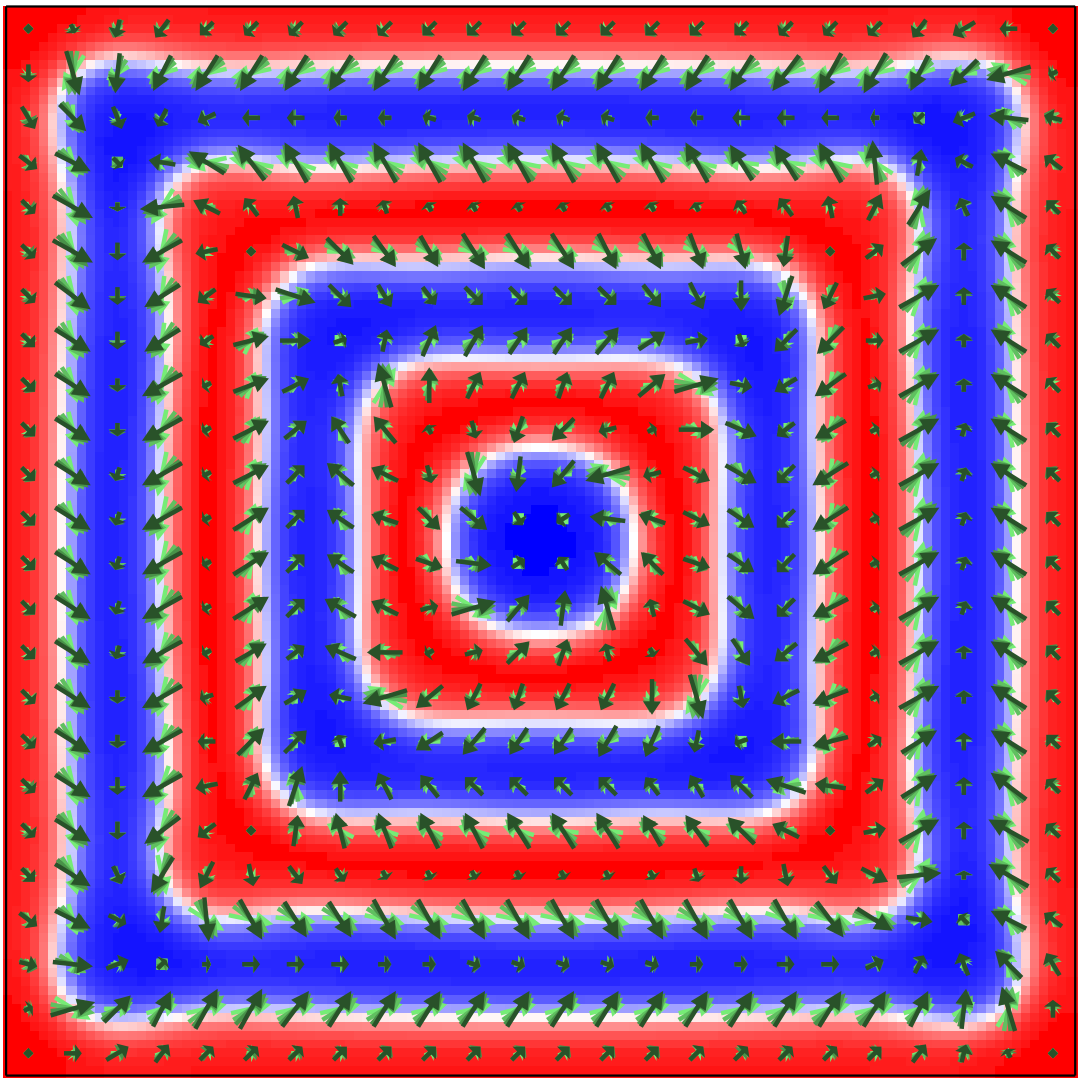
Final Magnetization Configuration

Simple Anisotropy

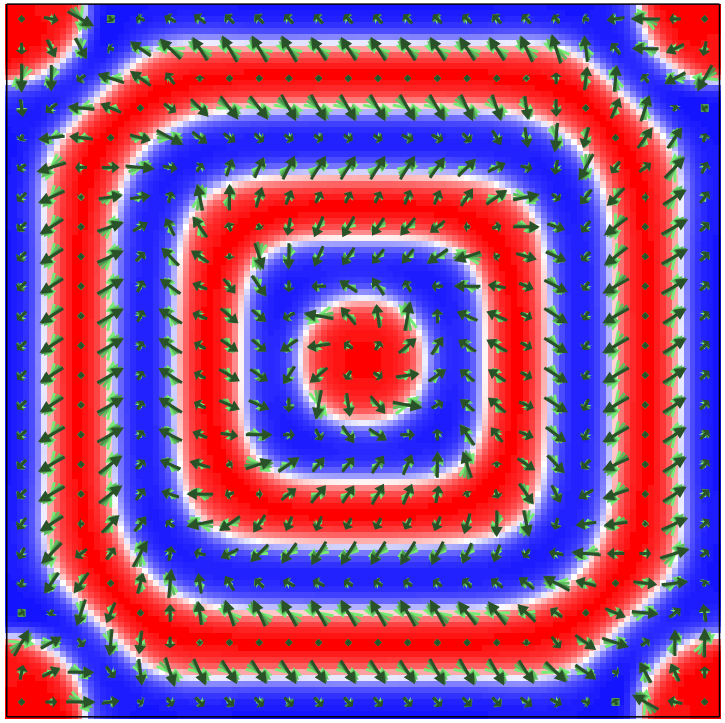


Final Magnetization Configuration

Extended Anisotropy

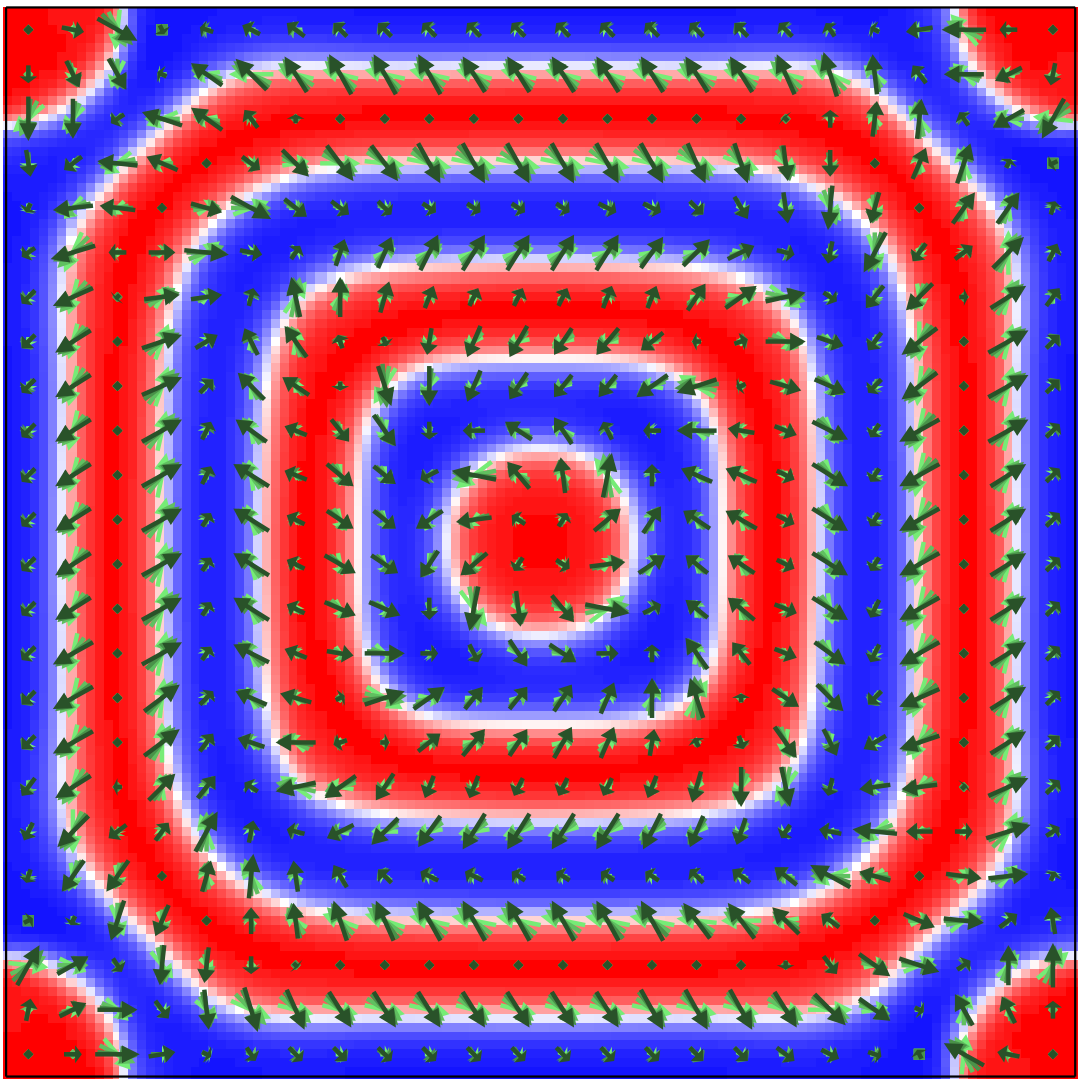


Final Magnetization Configuration



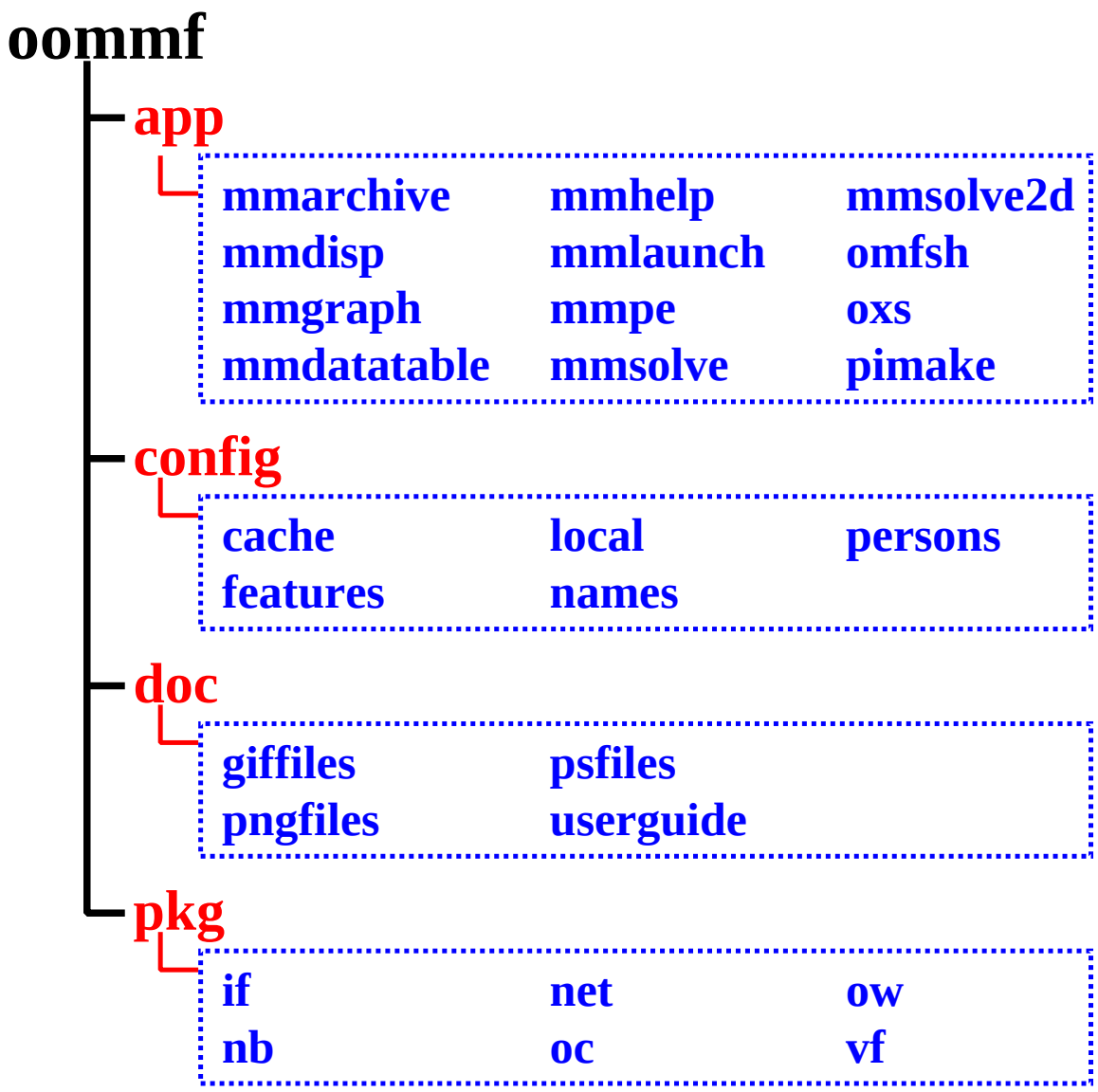
Extended anisotropy, box initial state.

Final Magnetization Configuration

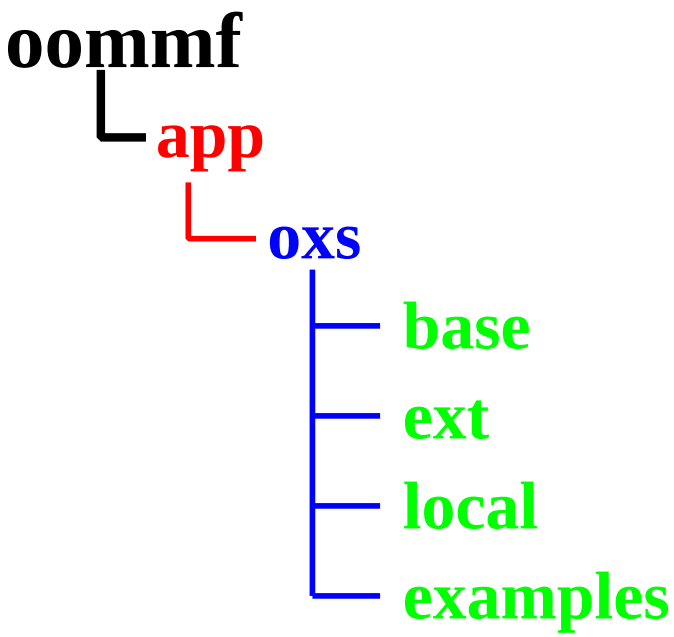


Extended anisotropy, box initial state.

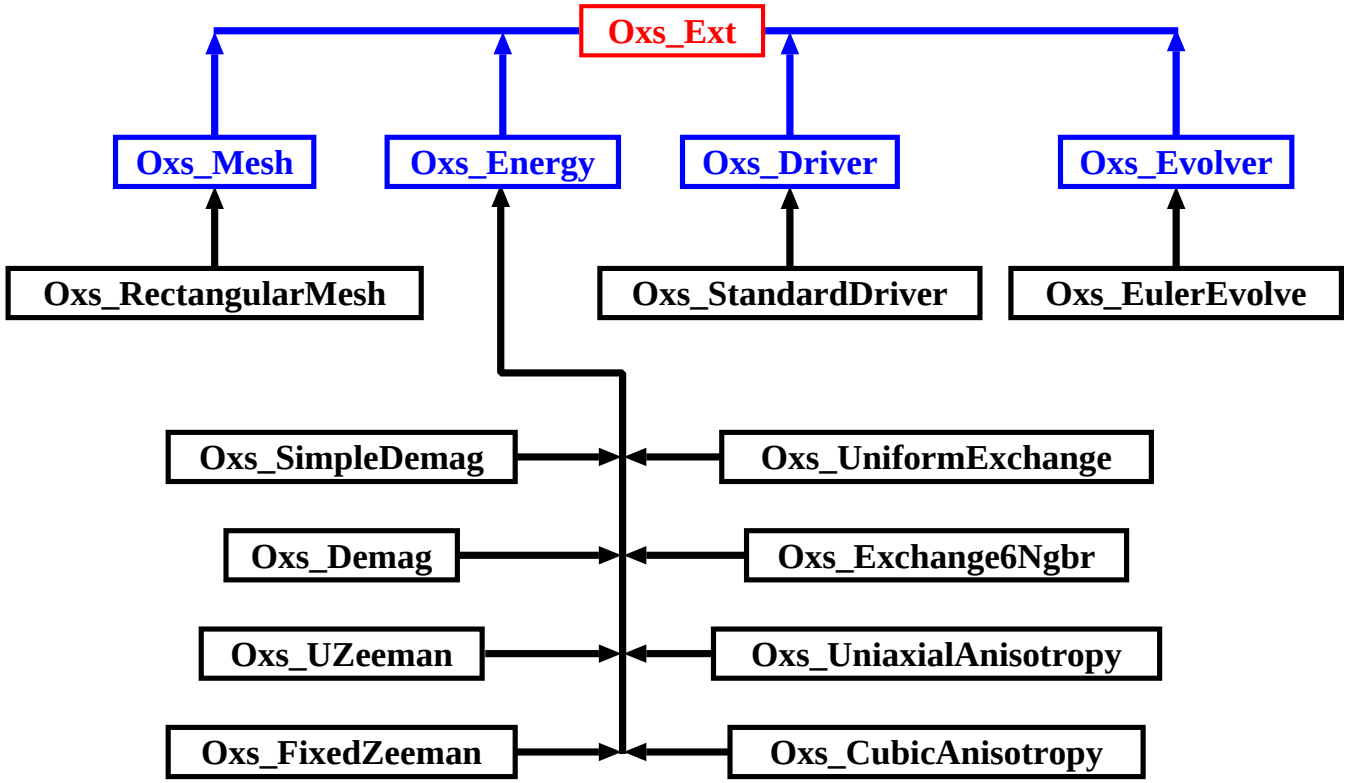
OOMMF Directory Layout



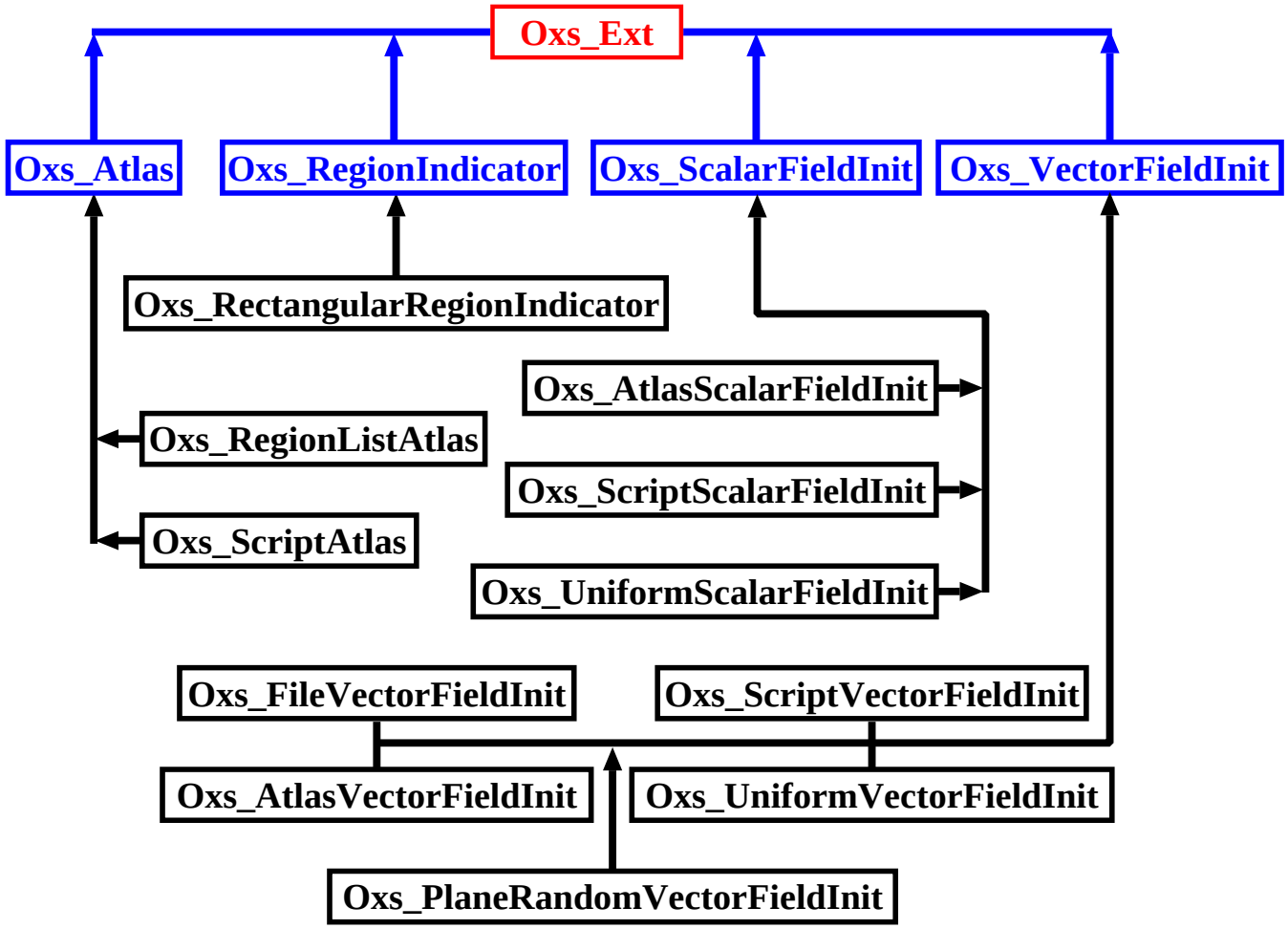
OXS Subdirectory Layout



Oxs_Ext Main Tree



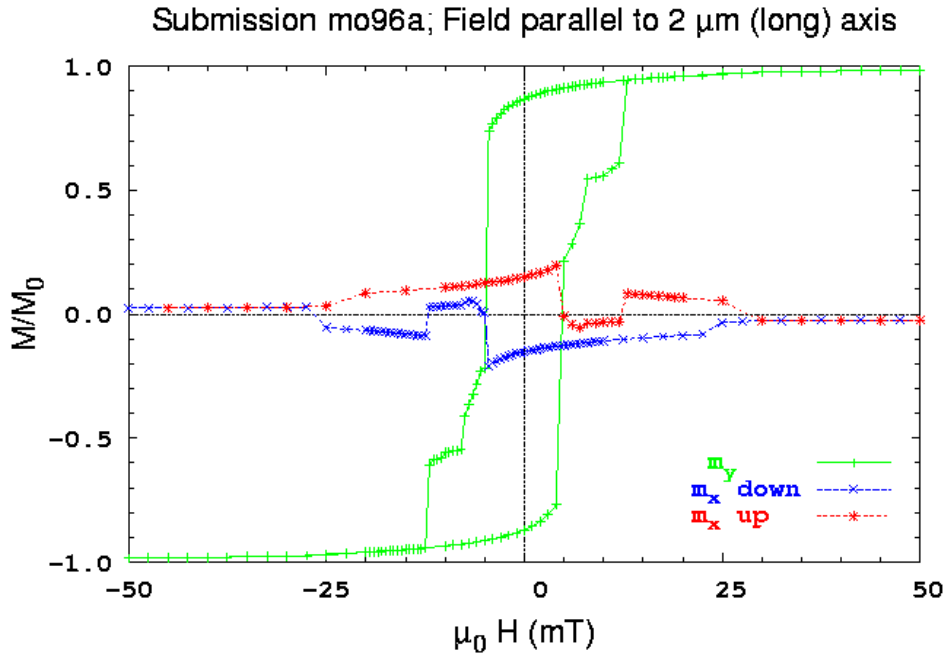
Oxs_Ext Support Tree



Testbed Systems

Platform	Compilers
AIX	VisualAge C++ (xlC), Gnu gcc
Alpha/Compaq Tru64 UNIX	Compaq C++, Gnu gcc
Alpha/Linux	Compaq C++, Gnu gcc
Alpha/Windows NT	Microsoft Visual C++
HP-UX	aC++
Intel/Linux	Gnu gcc
Intel/Windows NT, 95, 98	Microsoft Visual C++, Cygwin gcc, Borland C++
MIPS/IRIX 6 (SGI)	MIPSpro C++, Gnu gcc
SPARC/Solaris	Sun Workshop C++, Gnu gcc

Hysteresis Loop Calculations




```
FOR i = 1 to N
    Apply external field i
    WHILE(not equilibrium)
        Take time step
        Calculate energies and fields
    END WHILE(not equilibrium)
END FOR i
```

Cell-Based Calculations

```
FOR cell = 1 to N
   FOR energy = 1 to M
      cell->CalculateEnergy[energy]
   END FOR energy
END FOR cell
```

Energy-Based Calculations

```
FOR energy = 1 to M
   FOR cell = 1 to N
      energy->CalculateEnergy[cell]
   END FOR cell
END FOR energy
```

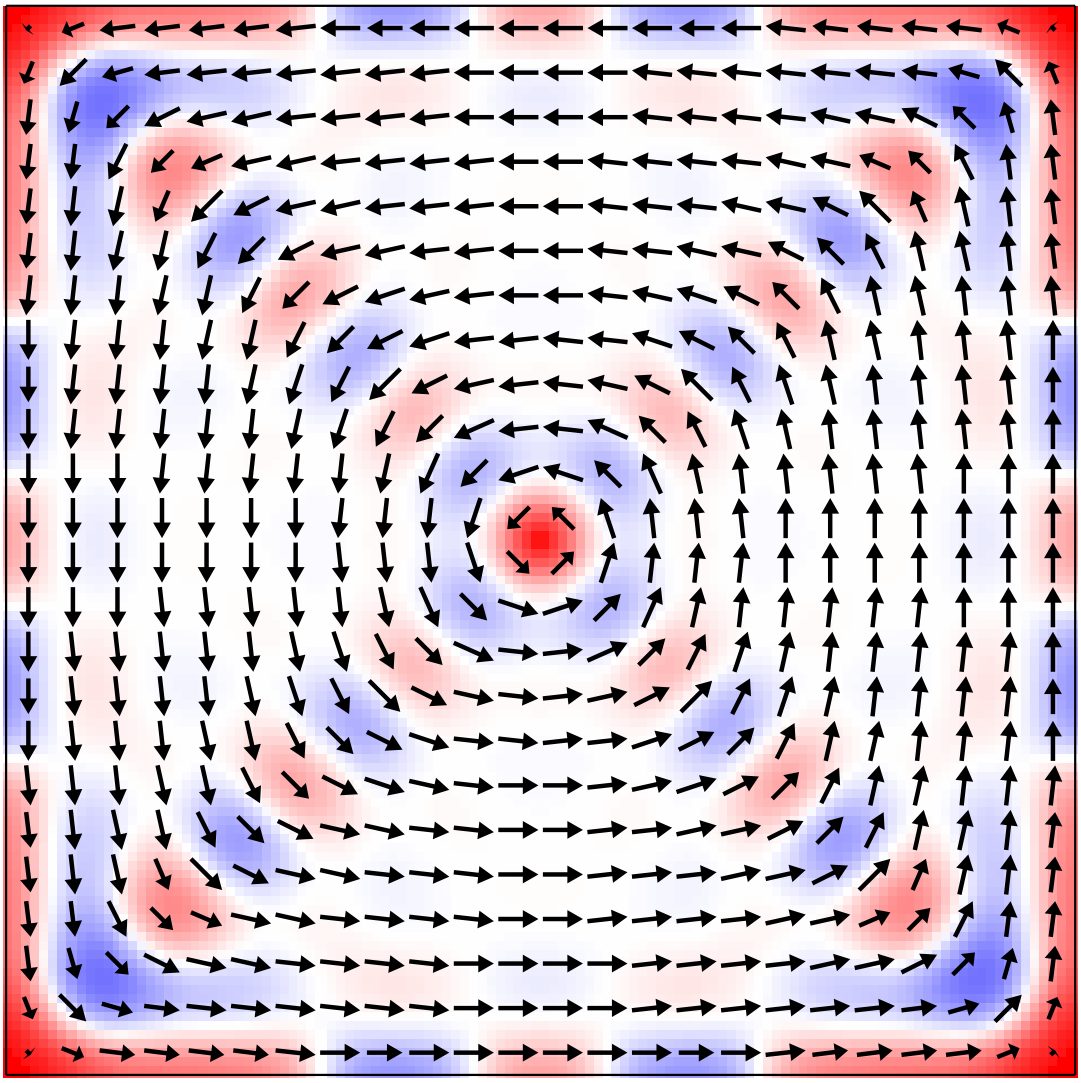
Advantages to Energy-Based Approach

1. Encapsulation of material parameters
2. Efficient demag calculation
3. Typical output requirements
4. Expectations of end users and extension writers

Disadvantages?

- Exposure of mesh details
- Shared material parameters
- Multiple spin array traversals

Final Magnetization Configuration



Simple anisotropy, 2D model

Inital state: 3D model final state