An Introduction to the OOMMF eXtensible Solver Class Architecture

M. J. Donahue and D. G. Porter

NIST, Gaithersburg, MD USA

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

1901-2001
NIST CENTENNIAL

The

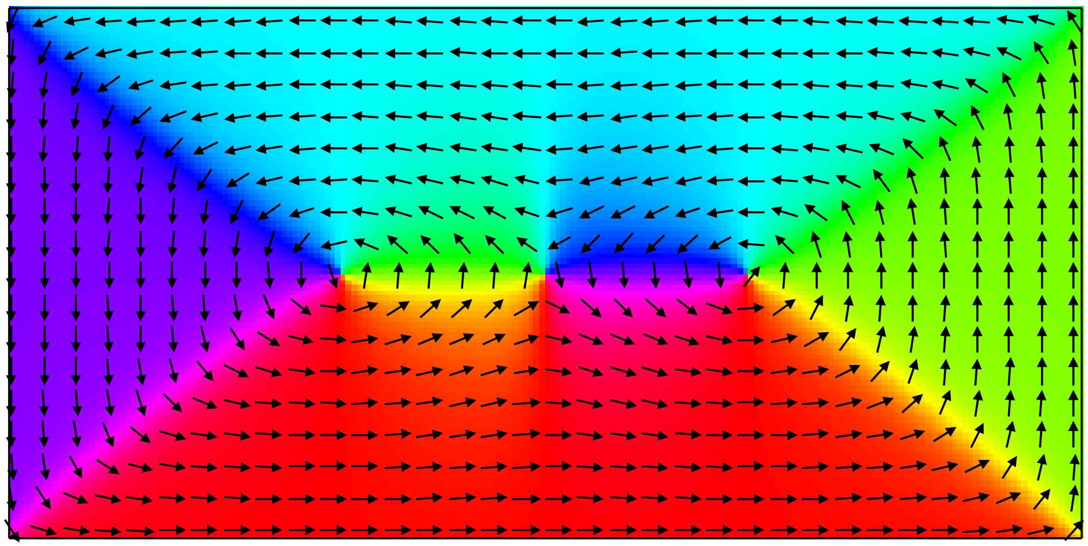
OOMMF

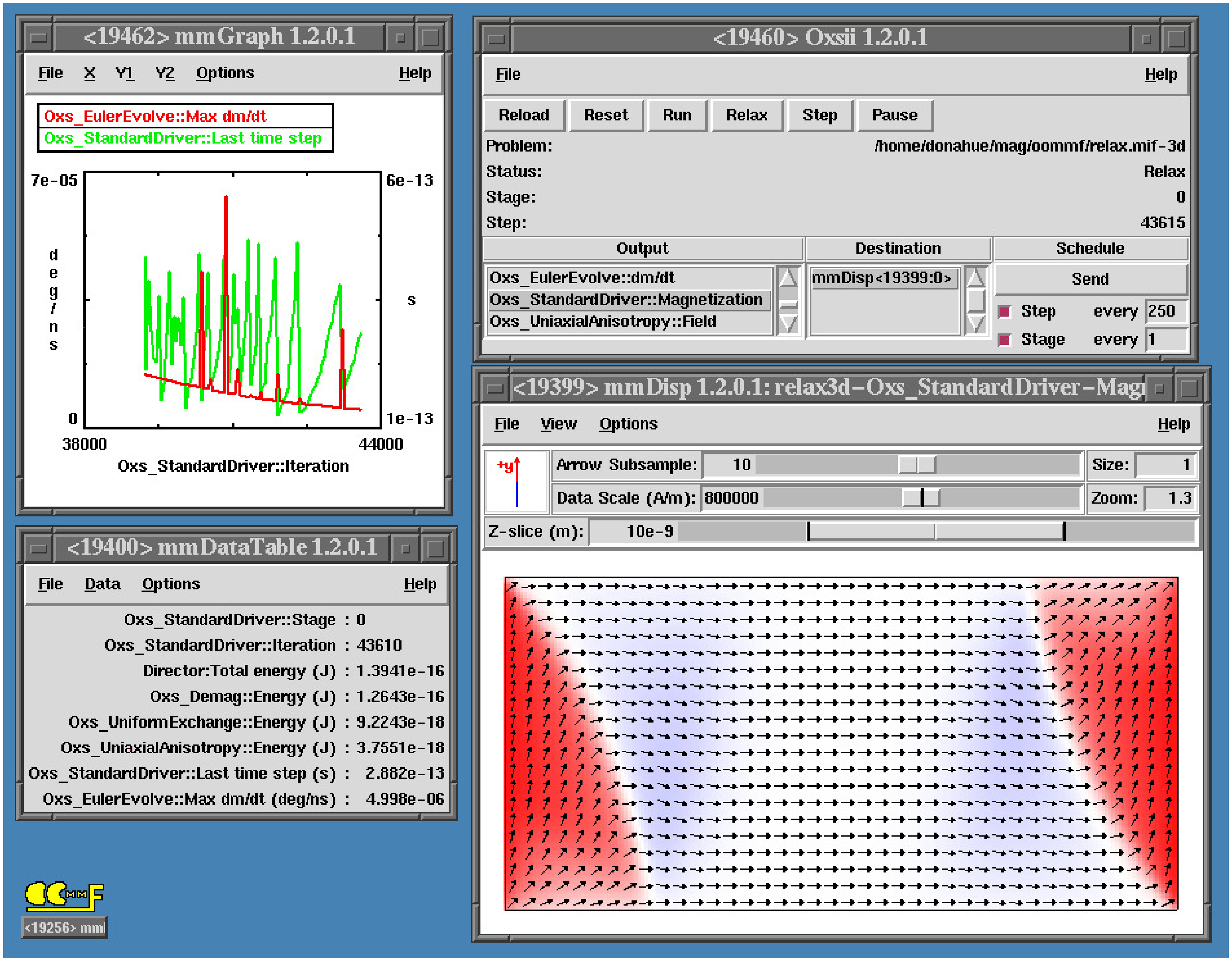
eXtensible

Solver

Remanent Magnetization

1 μm $\times$ 0.5 μm $\times$ 5 nm





<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-Mag
File View Options Help
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> mml

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

Micromagnetic Equations

Landau-Lifshitz-Gilbert:

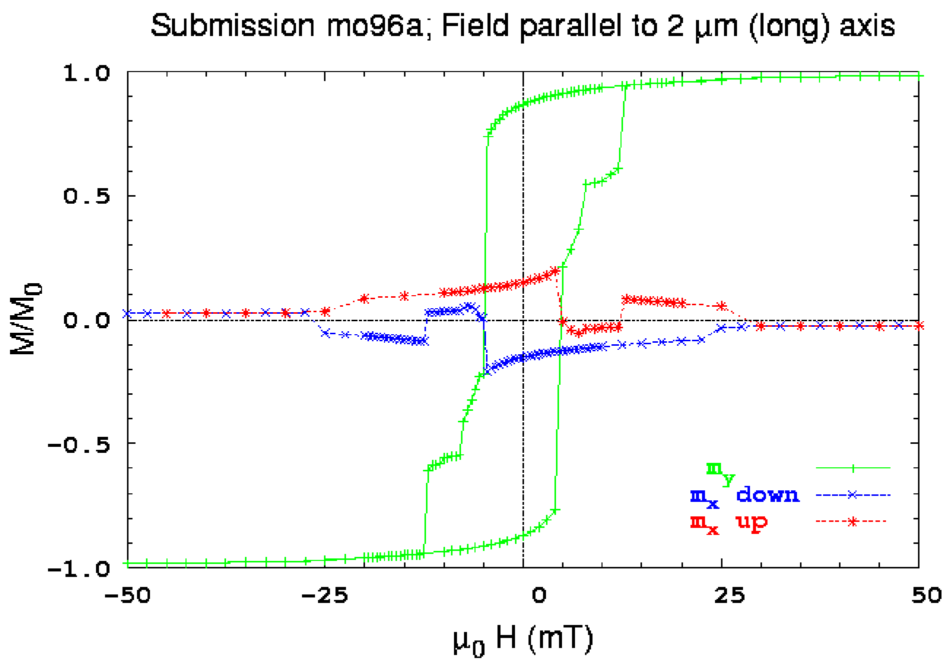
$$\frac{d\mathbf{M}}{dt} = \frac{-\omega}{1+\lambda^2}\mathbf{M}\times\mathbf{H}_{\text{eff}} - \frac{\lambda\,\omega}{(1+\lambda^2)M_{\text{s}}}\mathbf{M}\times(\mathbf{M}\times\mathbf{H}_{\text{eff}})$$

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

Energies:

$$\begin{aligned}
E_{\text{exchange}} &= \frac{A}{M_s^2}\left(|\nabla M_x|^2 + |\nabla M_y|^2 + |\nabla M_z|^2\right) \\
E_{\text{anis}} &= \frac{K_1}{M_s^4}(M_x^2 M_y^2 + M_y^2 M_z^2 + M_z^2 M_x^2) \\
E_{\text{demag}} &= \frac{\mu_0}{8\pi}\mathbf{M}(r)\cdot\left[\int_V \nabla\cdot\mathbf{M}(\mathbf{r}')\frac{\mathbf{r}-\mathbf{r}'}{|\mathbf{r}-\mathbf{r}'|^3}\,d^3r' \right. \\
&\qquad \left. - \int_S \hat{\mathbf{n}}\cdot\mathbf{M}(\mathbf{r}')\frac{\mathbf{r}-\mathbf{r}'}{|\mathbf{r}-\mathbf{r}'|^3}\,d^2r'\right] \\
E_{\text{Zeeman}} &= -\mu_0\mathbf{M}\cdot\mathbf{H}_{\text{ext}}
\end{aligned}$$

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

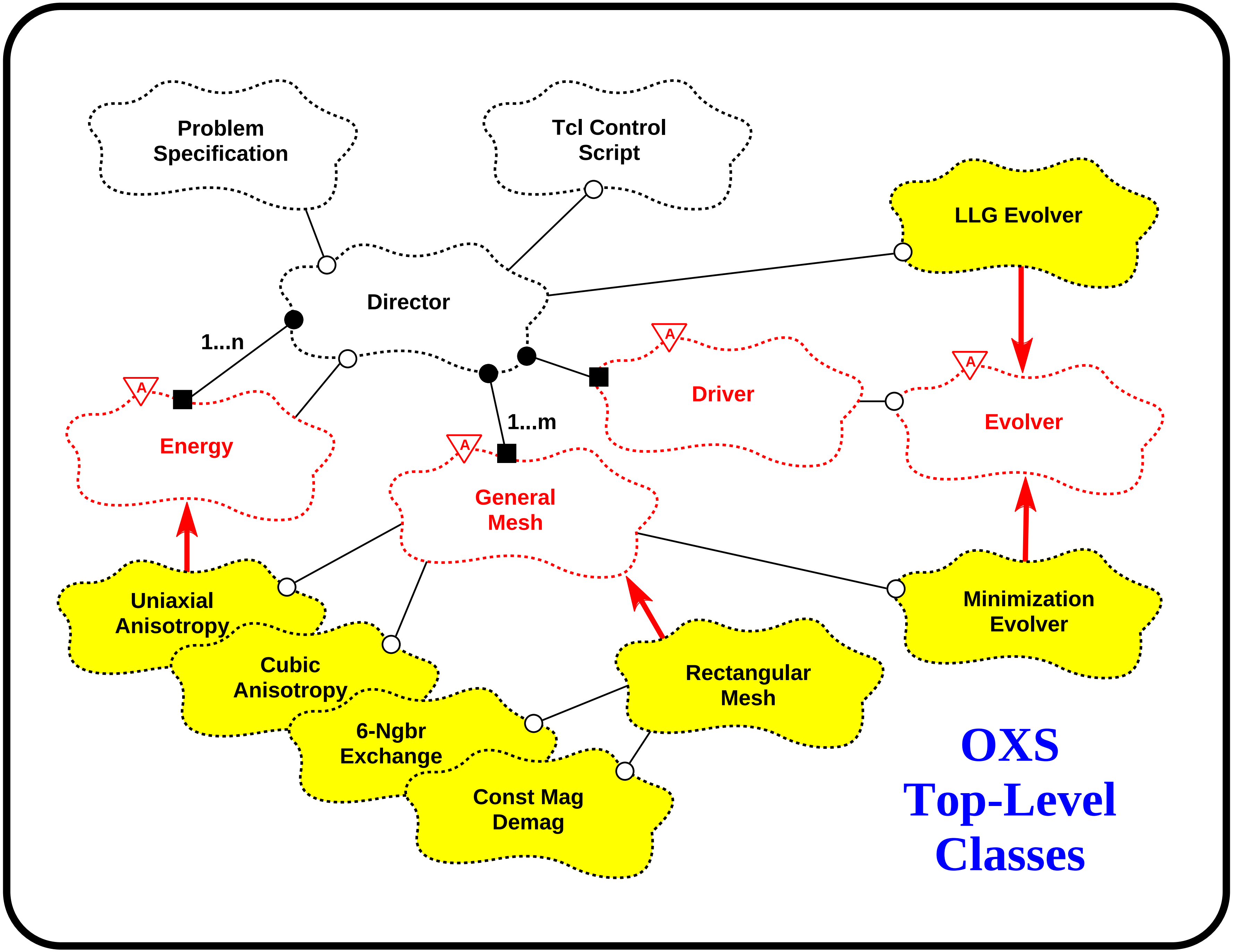
1. Encapsulation of material parameters
2. Efficient demag calculation
3. Typical output requirements
4. Expectations of end users and extension writers

Disadvantages?

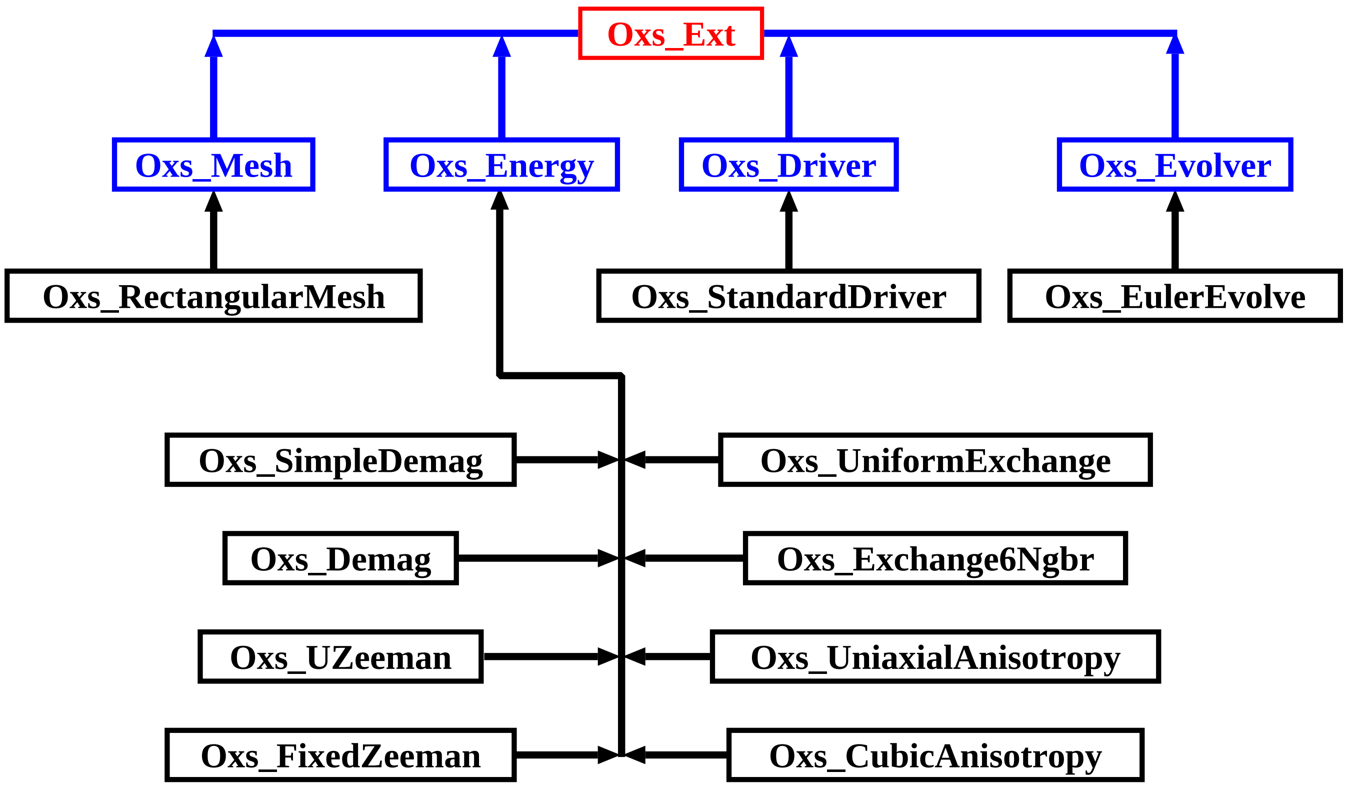
- Exposure of mesh details
- Shared material parameters
- Multiple spin array traversals

Mesh Downcasting in Oxs_Demag

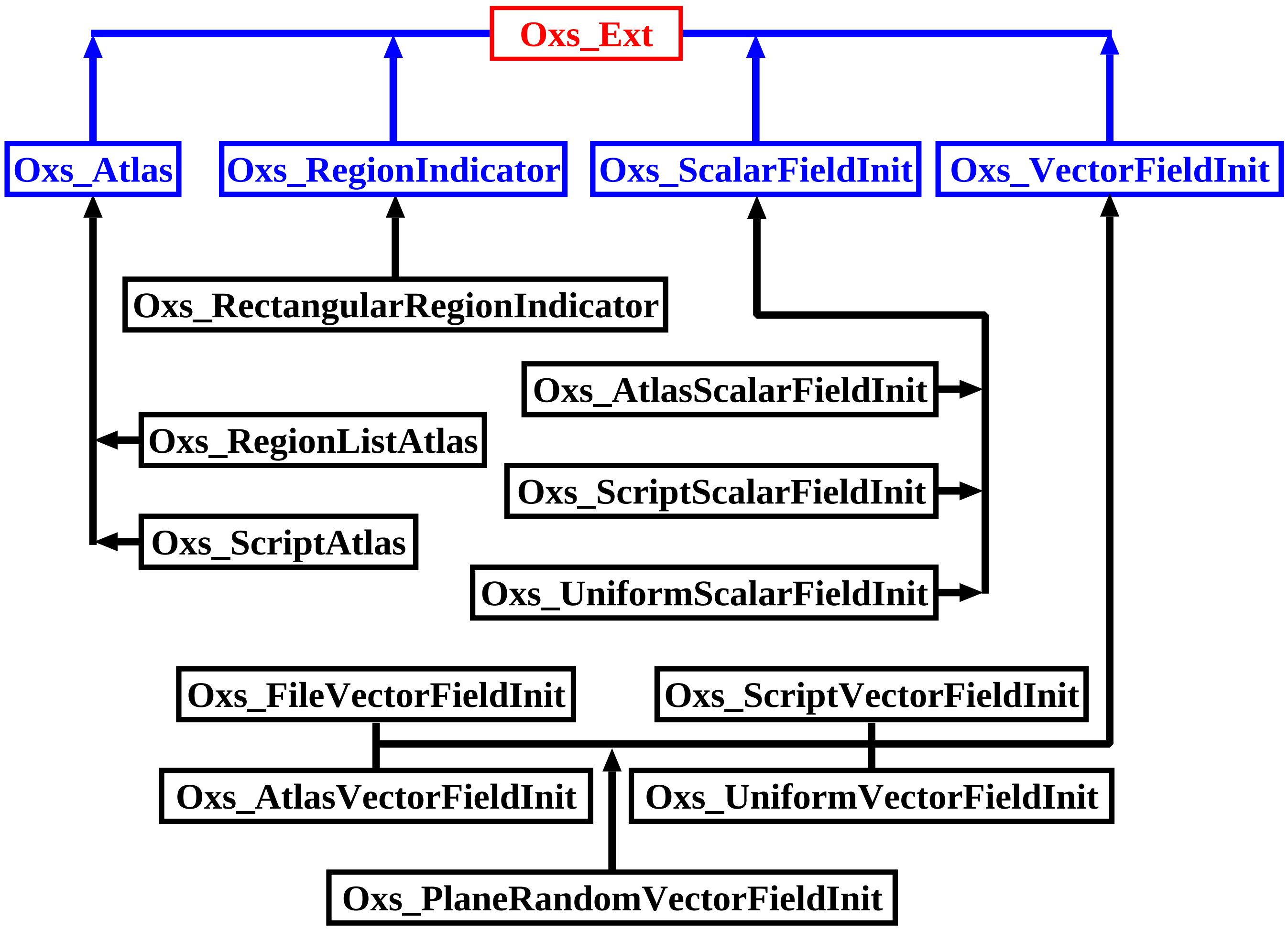
```
void Oxs_Demag::FillCoefficientArrays
                (const Oxs_Mesh* genmesh) const
{ // This routine is conceptually const.
  const Oxs_RectangularMesh* mesh
    = dynamic_cast<const Oxs_RectangularMesh*>(genmesh);
  if(mesh==NULL) {
    string msg=string("Object ")
      + string(genmesh->InstanceName())
      + string(" is not a rectangular mesh.");
    throw Oxs_Ext::Error(msg.c_str());
  }

...
```


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

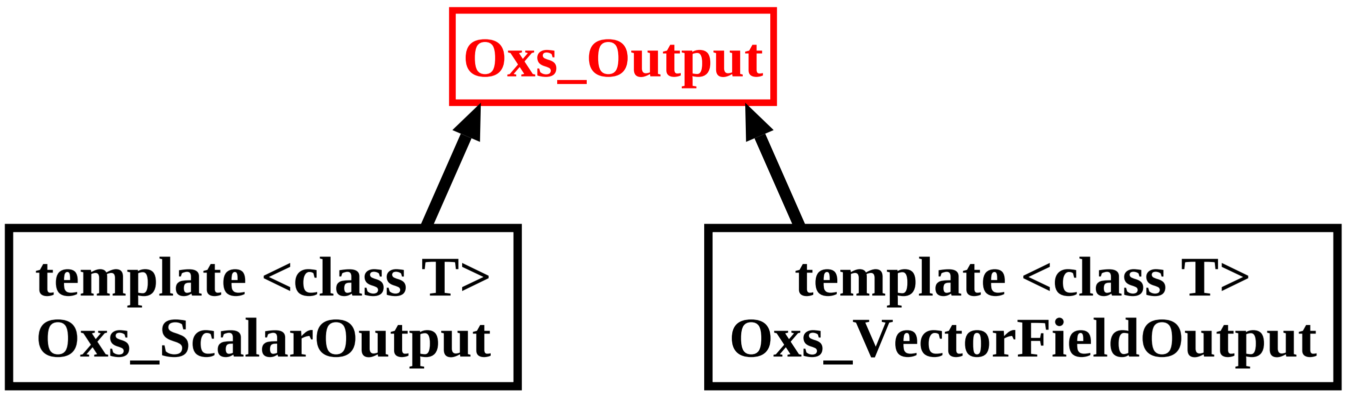
Oxs_Ext Main Tree



Oxs_Ext Support Tree



Oxs_Output Hierarchy



Sample MIF 2.0 File

```
# MIF 2.0

Specify Oxs_SimpleDemag {}

Specify Oxs_UniformExchange:NiFe {
  A  13e-12
}

Specify Oxs_EulerEvolve {
  alpha 0.5
  start_dm 0.01
}
```

```
Specify Oxs_SectionAtlas:atlas {
   world { Oxs_RectangularSection {
     xrange {1e-9 301e-9}
     yrange {0 200e-9}
     zrange {-1d-9 19e-9}
   }}
}

Specify Oxs_RectangularMesh:mesh {
  cellsize {10e-9 10e-9 10e-9}
  atlas :atlas
}
```

```
Specify Oxs_UniaxialAnisotropy {
  K1 { Oxs_UniformScalarFieldInit { value  6.2831853e4 } }
  axis { Oxs_RandomVectorFieldInit {
     min_norm 1
     max_norm 1
  } }
}
```

```
Specify Oxs_StandardDriver {
 evolver Oxs_EulerEvolve
 min_timestep 1e-18
 max_timestep 1e-9
 stopping_dm_dt 0.01
 mesh :mesh
 Ms { Oxs_UniformScalarFieldInit { value 8e5 } }
 m0 { Oxs_ScriptVectorFieldInit {
      script  {SineSpin 5}
      norm  1
 } }
}
```

```
proc SineSpin { freq x y z xmin ymin
                zmin xmax ymax zmax } {
    global pi
    set xspan [expr $xmax - $xmin]
    set mult [expr 2*$pi*$freq/$xspan]
    set vx [expr sin($mult*$x)]
    set vy [expr cos($mult*$x)]
    set vz 0
    return "$vx $vy $vz"
}
```

Sample Header File

```
/* FILE: simpleanisotropy.h
 *
 * Simple uniaxial anisotropy, derived from Oxs_Energy class.
 *
 */

#ifndef _OXS_SIMPLEANISOTROPY
#define _OXS_SIMPLEANISOTROPY

#include "energy.h"
#include "meshvalue.h"
#include "simstate.h"
#include "threevector.h"

/* End includes */
```

```
class Oxs_SimpleAnisotropy:public Oxs_Energy {
private:
  REAL8m K1;
  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
       Oxs_Director* newdtr, // App director
       Tcl_Interp* safe_interp, // Safe interpreter
       const char* argstr);  // MIF input block parameters
  virtual ~Oxs_SimpleAnisotropy() {}
  virtual void GetEnergyAndField(const Oxs_SimState& state,
                  Oxs_MeshValue<REAL8m>& energy,
                  Oxs_MeshValue<ThreeVector>& field) const;
};
#endif // _OXS_SIMPLEANISOTROPY
```

Sample Source Code File

```
// FILE: simpleanisotropy.cc

#include "nb.h"
#include "simpleanisotropy.h"

// Oxs_Ext registration support
OXS_EXT_REGISTER(Oxs_SimpleAnisotropy);

/* End includes */
```

```
// Constructor
Oxs_SimpleAnisotropy::Oxs_SimpleAnisotropy(
  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 initialization string
  K1 = GetRealInitValue("K1");
  axis = GetThreeVectorInitValue("axis");
  axis.SetMag(1.0);
  VerifyAllInitArgsUsed();
}
```

```
void Oxs_SimpleAnisotropy::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();

  REAL8m field_mult = (2.0/MU0)*K1;
  for(UINT4m i=0;i<size;++i) {
    REAL8m dot = axis*spin[i];
    field[i] = (dot*field_mult*Ms_inverse[i]) * axis;
    energy[i] = -K1*dot*dot;
  }
}
```

Sample Specify Block

```
Specify Oxs_SimpleAnisotropy {
  K1 530e3
  axis { 1 1 0 }
}
```

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 from OOMMF distribution!

Standard Cubic Anisotropy:

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

Extended Cubic Anisotropy:

$$E_{\text{anis}} = \frac{K_1}{M_s^4}\left(M_x^2 M_y^2 + M_y^2 M_z^2 + M_z^2 M_x^2\right) + \frac{K_2}{M_s^6}\left(M_x^2 M_y^2 M_z^2\right)$$

OOMMF Directory Layout

oommf

- **app**
  - mmarchive, mmhelp, mmsolve2d
  - mmdisp, mmlaunch, omfsh
  - mmgraph, mmpe, oxs
  - mmdatatable, mmsolve, pimake
- **config**
  - cache, local, persons
  - features, names
- **doc**
  - giffiles, psfiles
  - pngfiles, userguide
- **pkg**
  - if, net, ow
  - nb, oc, vf

OXS Subdirectory Layout

- **oommf**
  - **app**
    - **oxs**
      - **base**
      - **ext**
      - **local**
      - **examples**

Web Pages

- Home Page:

  `http://math.nist.gov/~MDonahue/`

- OOMMF:

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

- μMAG:

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