BornAgain
Simulate and fit neutron and x-ray scattering at grazing incidence

Céline Durniak, Gennady Pospelov, Walter Van Herck, Joachim Wuttke
April 9, 2013
Outline

- Motivation
- BornAgain history
- BornAgain usage
- Future
Motivation

From local (MLZ) and more broader needs (e.g. HDRI):
- Start community project for different user experiments (neutron and x-ray grazing incidence)
- More generally structured than existing software
- IsGISAXS as reference software
Outline

- Motivation
- BornAgain history
- BornAgain usage
- Future
Background

IsGISAXS as a starting point:

- Very successful software which is a de facto standard in the user community

- No longer actively supported
Starting

New software platform or extend existing?

Extending IsGISAXS:

- Code reuse

From scratch in C++:

- Developers background
- Sample structure
Development

- Comparison with standard: more easy
- IsGISAXS examples as milestones along the way
- OO model facilitates extension
- Agile development
Development sprints

Benefits:
- Short release cycles
- Code stability
<table>
<thead>
<tr>
<th>Status</th>
<th>Issue</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint 12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Related Issues:**
- **Bug #214:** Weird double delete somewhere in boost/gzip while running functional tests on mac=gcc48
- **Bug #220:** Gui: provide GUI compilation under Qt5/Linux
- **Bug #220:** PythonAPI: provide Py++ code generation under Linux
- **Bug #244:** OutputDataOfFactory: make consistent [x][y] axis ordering while writing/reading OutputData from *.txt and *.jna files
- **Bug #245:** Inconsistent fact 2*Pi in FormFactorHemiSpheroId
- **Bug #246:** PythonAPI: resolve bug with 32bit pointer type generated by Py++
- **Bug #247:** PythonAPI: kvector algebra dissapeared from python
- **Bug #254:** Problem with form factor of cylinder
- **Feature #460:** Framework general: implement simple logging tool with control of output level
- **Feature #492:** Provide BornAgainFlexCore with ROOT minimizer wrappers
- **Feature #498:** Provide python example of fitting using external ROOT
- **Feature #519:** Provide python script editing in GUI
- **Feature #526:** Building: prevent build failure in the case of ROOT is absent on the system
- **Support and Documentation #157:** Update NS Theory documents
- **Support and Documentation #166:** Create architectural overview documentation
- **Support and Documentation #199:** Doxygen comments for all classes
- **Support and Documentation #202:** Write description of logisaxx01 example for User Manual using latex
- **Support and Documentation #250:** Provide corrected "number of line of code vs time" script and plot
- **Support and Documentation #257:** Make tarball BornAgain.zip for the Workshop
- **Refactoring #204:** Implement ISampleVisitor
- **Refactoring #222:** Get rid from Instance() method for all singletons
- **Refactoring #222:** Check replacement of Coordinate3D<INT> by Geometry::BasicVector3D<INT>
- **Refactoring #238:** Restructure Examples directory
- **Refactoring #248:** PythonAPI: redesign Py++ boost::python API code generation and installation
- **Testing #209:** Run valgrind on all FunctionalTests
- **Testing #228:** Add mesocrystal test in FunctionalTestCor and TestPyCore
- **Testing #233:** PythonAPI: learn how to create shared_ptr objects in python and deliver them into C++
- **Testing #255:** Repair compilation under JCNS linux, OpenSuse 12.3
Basic software architecture

- User program/script
- Core
  - Samples and algorithms
  - GSL
  - Boost
- Fitting
- Graphics
  - (only C++)
- External graphics
  - (e.g. matplotlib)
Current status

- Beta
- Transition to production release with users’ help
- Everyone is welcome to try it out, when their experiment requires extension of IsGISAXS
Supported sample structures

- Multilayer
- Interface roughness
- Multiple nanoparticles (shapes, densities)
- Interference functions
- Nanoparticles assemblies
Outline

- Motivation
- BornAgain history
- BornAgain usage
- Future
GISAS Geometry
Example 1

- User program in C++
- Core Samples and algorithms
- Graphics (only C++)
Example 1

Single cylinder formfactor
Example 1

```cpp
// IsGISAXS3 functional test: cylinder on the substrate
void FunctionalTests::IsGISAXS03::runDWBA()
{
    // building sample
    MultiLayer multi_layer;
    const IMaterial *p_air_material = MaterialManager::getHomogeneousMaterial("Air", 1.0, 0.0);
    const IMaterial *p_substrate_material = MaterialManager::getHomogeneousMaterial("Substrate", 1.0-6e-6, 2e-8);
    Layer air_layer;
    air_layer.setMaterial(p_air_material);
    Layer substrate_layer;
    substrate_layer.setMaterial(p_substrate_material);
    complex_t n_particle(1.0-6e-4, 2e-8);
    ParticleDecoration particle_decoration(new Particle(n_particle, new FormFactorCylinder(5*Units::nanometer, 5*Units::nanometer)));
    particle_decoration.addInterferenceFunction(new InterferenceFunctionNone);
    LayerDecorator air_layer_decorator(air_layer, particle_decoration);
    multi_layer.addLayer(air_layer_decorator);
    multi_layer.addSubstrate(substrate_layer);

    // building simulation
    Simulation simulation;
    simulation.setDetectorParameters(100, 0.0*Units::degree, 2.0*Units::degree,
                                       100, 0.0*Units::degree, 2.0*Units::degree, true);
    simulation.setBeamParameters(1.0*Units::angstrom, -0.2*Units::degree, 0.0*Units::degree);

    // running simulation and copying data
    simulation.setSample(multi_layer);
    simulation.runSimulation();
    m_results[kTest_DWBA] = simulation.getOutputDataClone();
}
```
Example 2

- Python script
- Python wrappers
- Core (C++)
  - Samples and algorithms
Example 2

Isotropic hexagonal 2D paracrystal interference function
```python
def RunSimulation2():
    # defining materials
    mAmbience = MaterialManager.getHomogeneousMaterial("Air", 1.0, 0.0)
    mSubstrate = MaterialManager.getHomogeneousMaterial("Substrate", 1.0-6e-6, 2e-8)
    # collection of particles
    n_particle = complex(1.0-6e-4, 2e-8)

cylinder_ff = FormFactorCylinder(5*nanometer, 5*nanometer)
cylinder = Particle(n_particle, cylinder_ff)

interference = InterferenceFunction2DParaCrystal.createHexagonal(20.0*nanometer, 0.0, 20.0*micrometer, 20.0*micrometer)
pdf = FTDistribution2DCauchy(1.0*nanometer, 1.0*nanometer)
interference.setProbabilityDistributions(pdf, pdf)

particle_decoration = ParticleDecoration()
particle_decoration.addParticle(cylinder, 0.0, 1.0)
particle_decoration.addInterferenceFunction(interference)

air_layer = Layer(mAmbience)
air_layer_decorator = LayerDecorator(air_layer, particle_decoration)
substrate_layer = Layer(mSubstrate, 0)

multi_layer = MultiLayer()
multi_layer.addLayer(air_layer_decorator)
multi_layer.addLayer(substrate_layer)
# build and run experiment
simulation = Simulation()
simulation.setDetectorParameters(100, 0.0*degree, 2.0*degree, 100, 0.0*degree, 2.0*degree, True)
simulation.setBeamParameters(1.0*angstrom, -0.2*degree, 0.0*degree)
simulation.setSample(multi_layer)
simulation.runSimulation()
return GetOutputData(simulation)
```
Example 3: mesocrystals
Example 3: mesocrystal formfactor

\[ S_{MC}(r) = S_{cyl}(r) \cdot \sum_{\{R_i\}} S_{NP}(r) \otimes \delta(r - R_i) \]

\[ F_{MC}(q) = F_{cyl}(q) \otimes \sum_{\{Q_i\}} F_{NP}(q) \cdot \delta(q - Q_i) \]

\[ = \sum_{\{Q_i\}} F_{cyl}(q - Q_i) \cdot F_{NP}(Q_i) \]
Example 3: mesocrystals

Real data (courtesy E. Josten)

12 parameter fit (with E. Josten, A. Glavic)

\[ \hat{\phi}_f (\text{radians}) \]

\[ \hat{\phi}_f (\text{radians}) \]
“Example 4”: prototype GUI

Development of Qt Graphical Interface:

Instrument definition

Graphical output
Outline

- Motivation
- BornAgain history
- BornAgain usage
- Future
Future

- Neutron polarization
- GUI (prototype under development)
- GPU computing
- Input from user community and collaborations
  http://apps.jcns.fz-juelich.de/doku/sc/bornagain:start
- Discussion tomorrow!
Design choices

- Flexible Architecture
- Open-source Platform-independent
- Support Maintenance
- Physics

SW development Practices

DWBA Sample models
Software development

General practices:
- Agile development
- Software tools for maintenance and support
- Assure code stability and quality

BornAgain specifics:
- C++ with Python API: fast and extendable
- Well established libraries: Boost, fftw3, etc.
- ROOT data analysis framework
- Qt for GUI development
Software development: general

Agile development

Benefits:
- Short release cycles
- Code stability
Software development: general

Software Tools

- Bug/Issue tracking: Redmine
- Version Control: git
- Nightly build server: TeamCity
- Code documentation: Doxygen
- Code browser: OpenGrok
Software development: general

Code stability and quality

• Code review and refactoring
• Unit tests: Googletest
• Functional tests: C++ and Python
Software development: BornAgain

Programming language
- C++: speed, extensibility, maintenance
- Python API: extensibility, customization by users

Libraries
- Boost, GSL, fftw3

ROOT data analysis framework
- Standard in High Energy Physics community
- Large user base
- Scientific plotting, fitting engines, etc.

(Qt for GUI development)
BornAgain

- Development started in April 2012
- Name refers to central role of DWBA
- Simulation and fitting of grazing incidence scattering for multilayered samples with embedded nanoparticles
- Core library: 25k lines of code
- GUI at prototype stage
Physical modelling

Distorted Wave Born Approximation
Physical modelling

Sample models

- Multilayer
- Interface roughness
- Multiple nanoparticles (shapes, densities)
- Interference functions
- Nanoparticles assemblies