BornAgain Tutorial

Jan Burle, Jonathan M. Fisher, Marina Ganeva, Gennady Pospelov, Walter Van Herck and Joachim Wuttke
Overview

- Theoretical background
- Experimental setup: beam, sample, detector (binning)
- Software API architecture
- Coordinate systems
- Development infrastructure
- GUI demo
- Python usage
Experimental setup
GISAS specifics

- In-plane and out-of-plane information
- Increased scattering volume
- Reflected and transmitted waves interfere
- Software can help to arrive at quantitative results (simulate & fit)

\[ V_{\text{scat}} \sim 2 \frac{A_{\text{beam}} \cdot t}{\sin \alpha_i} \]
Distorted Wave Born Approximation

- Unperturbed system: smooth multilayer
  Solutions: reflected and transmitted plane waves
- 1st order scattering on embedded nanoparticles: 4 terms
- Similar with roughness
Shape and position

- Shape: form factor
- Positions: interference function

\[ N \frac{d\sigma}{d\Omega} = \left| \sum_i \exp(iq \cdot R_i) F_{DWBA}(q; T_i) \right|^2 \]
Beam, detector

- Idealized beam: monochromatic, no angular divergence
  Real beam: model with distributions / resolution function
- Spherical or rectangular detector
Intensity in detector pixel

- Integration of differential cross-section
- If pixel is small enough, value at center of pixel times its size

\[ I_{\text{pixel}} = \int_{A_{\text{pixel}}} d\Omega \frac{I_0}{\sin \alpha_i} \rho_s \frac{d\sigma}{d\Omega} \]

\[ \approx \frac{I_0}{\sin \alpha_i} \rho_s \frac{d\sigma}{d\Omega} \Delta\Omega_{\text{pixel}} \]
Structure of software API: simulation

- GISASSimulation contains sample and instrument
- Additionally, it may contain distributions, builders, ...
- runSimulation calculates scattering intensity
- getIntensityData retrieves the intensity map

```
GISASSimulation
+ setSample
+ setInstrument
+ runSimulation
+ getIntensityData

MultiLayer
+ addLayer

Instrument
+ setBeam
+ setDetector
```
Structure of software API: sample tree

- Root of the tree is a multilayer
- Multiple layers possible
- Multiple layouts possible
- Multiple particles possible...

```
MultiLayer
  + addLayer

Layer
  + addLayout

Layer
  + addLayout

ParticleLayout
  + addParticle
  + addInterferenceF

Particle
  + addLayer

Particle
  + addLayer

IFLattice
  + addLayer
```
Structure of software API: instrument

- Instrument contains beam and detector
- Beam: wavelength, direction, intensity, polarization
- Detector: type, size, position, masks, ROI, ...

```
Instrument
+ setBeam
+ setDetector

Beam
+ setIntensity
+ setPolarization

Detector
+ setRegionofInter.
+ setDetectorAxes
```
Coordinate systems: global

- Global coordinate system
- z-axis pointing upwards
Coordinate systems: local

- Attached to each particle
- Rotations of particle are with respect to this local system
- Position of the particle is position of local origin in parent’s coordinate system
Coordinate systems: combined

- Particle’s z-position is relative to layer’s origin (top interface, except for top layer)
- This makes z negative for particles embedded in layers
Development infrastructure

- Source control: github
- Code review: github
- Continuous integration: github, buildbot
- Management of release cycles: Redmine
- Issue tracking: Redmine
- Unit testing: googletest
- Functional tests: ad hoc
- Documentation: drupal website & user manual
BornAgain demo

- **GUI**
  - Different screens
  - Defining the instrument
  - Building a sample
  - Simulating a sample
  - Simulate with MC
  - Export to python

- **Python**
  - Run simple script
Exercises

- Thin layer (20nm) on substrate, containing cubes (L=10nm) in the middle of the layer, with short range order (radial paracrystal; D=20nm); try also with rotating the cubes around z
- Multilayer system (repetition of bilayers: choose number of repetitions, layer thickness=5nm) with roughness (sigma=1nm, lateral correlation length=5nm) with and without cross-correlation
- Spheres (R=5nm) in hexagonal 2d lattice (D=10nm) on substrate