BornAgain: modeling and fitting GISAS
software architecture, project infrastructure

Céline Durniak, Marina Ganeva, Walter Van Herck,
Gennady Pospelov, Joachim Wuttke

Scientific Computing Group at MLZ
Jülich Center for Neutron Science

Erlangen GISAXS Workshop, May 6, 2015
Outline

- Motivation
- Software Architecture
- Functionality
- Validation
- Project Infrastructure
Generic framework to simulate and fit grazing-incidence small-angle scattering.

Physics model is based on the Distorted Wave Born Approximation

Development started in April, 2012.

Current release 1.1.0

Open source, GPL3 license
Motivation

1) Support for specific instruments at MLZ (Garching)
   - Serve our users, support in house research, at Maria and REFSANS instruments

2) HDRI (High Data Rate Processing and Analysis Initiative)
   - Start with grazing incidence small angle scattering
     - Create generic framework covering simulations of x-rays, polarized and non-polarized neutrons
     - Provide generic functionality/extensibility for broader usage
## IsGISAXS as a starting point:

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

<table>
<thead>
<tr>
<th>Package</th>
<th>Application</th>
<th>Platform</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsGISAXS</td>
<td>Nanostructures on surfaces</td>
<td>Windows, Unix</td>
<td>GNU Public</td>
</tr>
<tr>
<td>FitGISAXS</td>
<td>Buried nanostructures</td>
<td>IgorPRO</td>
<td>GNU Public + IgorPRO</td>
</tr>
<tr>
<td>HipGISAXS</td>
<td>Buried nanostructures</td>
<td>Unix, HPC Computing</td>
<td>Berkeley, non-commercial</td>
</tr>
</tbody>
</table>

---

**IsGISAXS**: a program for grazing-incidence small-angle X-ray scattering analysis of supported islands

R Lazzari - Journal of Applied Crystallography, 2002 - scripts.iucr.org

This paper describes a Fortran program, IsGISAXS, for the simulation and analysis of grazing-incidence small-angle X-ray scattering (GISAXS) of islands supported on a substrate. As is usual in small-angle scattering of particles, the scattering cross section is...

Cited by 257  Related articles  All 7 versions  Cite

- Simulation in DWBA
- FORTRAN 90, 13k lines of code
- No longer actively supported

---

05
IsGISAXS limitations

- no support for multi layer systems
- no multiple types of particles in single layer
- no polarized neutrons and magnetic materials
- no parallelization
- usability

*IsGISAXS parameter file*

```plaintext
# Base filename
isgi_2-types-of-particles

# Framework Diffuse, Multilayer, Number of index slices, Polarization
DWBA DA 0 25 ss

# Beam Wavelength : Lambda(nm), WL_distribution, Sigma_WL/WL, WL_min(nm), WL_max(nm), nWL, xWL
  0.1 none 0.3 0.08 0.12 20 3

# Beam Alpha_i : Alpha_i(deg), Ai_distribution, Sigma_Ai(deg), Ai_min(deg), Ai_max(deg), nAi, xAi
  0.2 none 0.1 0.15 0.25 30 2

# Beam 2Theta_i : 2Theta_i(deg), Ti_distribution, Sigma_Ti(deg), Ti_min(deg), Ti_max(deg), nTi, XTi
  0. none 0.5 -0.5 0.5 10 2

# Substrate : n-delta_S, n-beta_S, Layer thickness(nm), n-delta_L, n-beta_L, RMS roughness(nm)
  6.E-06 2.e-8 0. 1.E-05 5.E-07 0.

# Particle : n-delta_I, n-beta_I, Depth(nm), n-delta_SH, n-beta_SH
  6.E-04 2.e-8 0 8.E-04 2.e-0
```
Strategy

Reproduce functionality of IsGISAXS

Extend it with most demanded features
- multilayers, particle assemblies, polarized neutrons and magnetic domains
- Usability

Build user community and follow their demands
- Motivation
- **Software Architecture**
- Functionality
- Validation
- Project Infrastructure
Requirements

Users ➜ Easy-to-use
Requirements

Users → Easy-to-use → GUI
Requirements

Users → Easy-to-use → GUI

Users → Availability
Users

Easy-to-use

Availability

Windows

Mac

Linux

Multi platform

GUI

Requirements
Users

Easy-to-use

Availability

Windows
Mac
Linux

GUI

Open source
No proprietary software
Easy-to-install
Multi platform
Requirements

- Users
  - Easy-to-use
    - Availability
      - Open source
      - No proprietary software
      - Easy-to-install
    - Multi platform
  - Advanced Users
Requirements

易用性

可用性

高级用户

复杂模型

可扩展性

GUI

开源

无专有软件

易于安装

多平台

面向对象，解耦

BornAgain框架
Users

Easy-to-use

Availability

Windows

Mac

Linux

Advanced Users

Complex models

Performance

Extensibility

GUI

Open source

No proprietary software

Easy-to-install

Multi platform

C++

OO, decoupling

BornAgain framework
Requirements

- Users
  - Easy-to-use
  - Availability
    - Open source
    - No proprietary software
    - Easy-to-install
    - Multi platform
- Advanced Users
  - Complex models
    - Performance
    - Extensibility
      - C++
      - OO, decoupling
      - GUI
Requirements

Users
- Easy-to-use
  - GUI
- Availability
  - Open source
  - No proprietary software
  - Easy-to-install
- Advanced Users
  - Windows
  - Mac
  - Linux
  - Clusters
- Complex models
  - Performance
  - Extensibility
  - Scripting
  - C++
  - OO, decoupling
  - Python
Software Architecture

- Open source framework written from scratch on C++, interfaced with Python
  - distributed under GPL3 license
- Multi platform
  - Unix flavors, source code
  - Windows, binary installer package
  - Mac OS, binary installer package
- Object oriented approach for sample description
Software Architecture

- Open source framework written from scratch on C++, interfaced with Python
  - distributed under GPL3 license
- Multi platform
  - Unix flavors, source code
  - Windows, binary installer package
  - Mac OS, binary installer package
- Object oriented approach for sample description

```python
# defining materials
m_air = HomogeneousMaterial("Air", 0.0, 0.0)
m_substrate = HomogeneousMaterial("Substrate", 6e-6, 2e-8)
m_particle = HomogeneousMaterial("Particle", 6e-4, 2e-8)

# collection of particles
cylinder_ff = FormFactorCylinder(5*nanometer, 5*nanometer)
cylinder = Particle(m_particle, cylinder_ff)
prism_ff = FormFactorPrism3(10*nanometer, 5*nanometer)
prism = Particle(m_particle, prism_ff)
particle_layout = ParticleLayout()
particle_layout.addParticle(cylinder, 0.0, 0.5)
particle_layout.addParticle(prism, 0.0, 0.5)

# air layer with particles and substrate form multi layer
air_layer = Layer(m_air)
air_layer.addLayer(particle_layout)
substrate_layer = Layer(m_substrate)
multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)
```
Software Architecture

Main components
- 110k lines of code
- Kernel contains two C++ libraries for simulation description and fitting
  - can be used from Python thanks to the boost-python bindings
- BornAgain GUI application
Software Architecture

External dependencies

- Well established Open Source libraries
  - Eigen, GSL, Boost, fftw3, Qt5
  - optionally ROOT from High Energy Physics community
Working with BornAgain

- Easy way
  - Using Graphical User Interface
Working with BornAgain

- Flexible way
  - Running Python script with simulation description
  - Visualizing results using external plotting tools

![Diagram showing the software architecture of BornAgain framework]

**Python script**
- User
- matplotlib
- BornAgain framework
  - Graphical User Interface
  - ROOT
  - Qt5
  - Eigen, GSL, Boost, fftw3

**Python bindings**
- libBornAgainCore: Samples and algorithms
- libBornAgainFit: Fitting routines
Software Architecture

Working with BornAgain

- Hard core way
  - Writing own C++ program linked with BornAgain libraries

```
C++ -> User
    |________________|
    | Python bindings|
    | libBornAgainCore|
    | Samples and algorithms |
    |________________|
    | Python bindings|
    | libBornAgainFit|
    | Fitting routines |
    |________________|
    | Eigen GSL Boost fftw3 |
```

```
    |________________|
    | Python bindings|
    |________________|
    | BornAgain |
    | Graphical User Interface |
    |________________|
    |________________|
    | ROOT |
    |________|
    | Qt5 |
```
Functionality

- X-rays, non-polarized and polarized neutrons
- Arbitrary number of layers
- Arbitrary number of different included particles (form factors, material)
- Arbitrary number of interference functions between the particles
- Inter-layer roughnessness (also with correlation)
- Nanoparticle assemblies
- Off-specular geometry, beam divergence, particle size distribution, detector resolution
- Motivation
- Software Architecture
- Functionality
- Validation
- Project Infrastructure
Validation against IsGISAXS

BornAgain results coincide with IsGISAXS on numerical level, except few geometry cases known to be buggy in IsGISAXS.

BornAgain framework

Erlangen GISAXS Workshop, May 6, 2015
BornAgain vs experiment: Ag nanoparticles on PPTE layer

Simulation of 3 layers system with Ag nanoparticles with broad size distribution

See M. Ganeva, talk at DPG 2015, Berlin
http://apps.jcns.fz-juelich.de/doku/sc/_media/dpg-berlin-talk1.pptx
BornAgain vs experiment: mesocrystals

Grazing incidence small-angle x-ray scattering from a mesocrystalline system
- mesocrystal consist of an FCC lattice composed of 5 nm spherical particles

Soleil beamline

BornAgain

Elisabeth Josten, PhD thesis
- Motivation
- Software Architecture
- Functionality
- Validation
- Project Infrastructure
Project infrastructure

Entry point
- bornagainproject.org

General practices
- Agile development
- Quality control

Software tools
- Bug/issue tracking: Redmine
- Version control: Git
- Nightly build server: TeamCity
- Code documentation: Doxygen
- Unit tests: Googletest
Welcome to BornAgain

BornAgain is a software package to simulate and fit small-angle scattering at grazing incidence. It supports analysis of both X-ray (GISAXS) and neutron (GISANS) data. Its name, BornAgain, indicates the central role of the distorted wave Born approximation in the physical description of the scattering process. The software provides a generic framework for modeling multilayer...
Agile development

- Workflow consists of sprint cycles every 4-6 weeks during which the team creates finished portions of the product.

---

**Roadmap**

**Sprint 24**

25 days running (15 Sep 2014)

User requests, toward beta of GUI in October.

- 15 closed (49%)
- 16 open (52%)

**Related issues**

- Bug #776: GUI: interference function 2D paraCrystal, rotation angle activation
- Bug #780: Windows: pixmap of item being dragged is not displayed on DesignerScene
- Bug #788: Fix release script to process CHANGELOG correctly
- Bug #791: LDataTest, DataAssignment, Unittest failure
- Bug #823: Remove interference function approximations from GUI
- Bug #826: cmake fails under Debian/testing: problem with Python
- Bug #829: CMake is not able to find right Python version when there is a Python2 and Python3
- Feature #393: Create Mac installer
- Feature #586: Investigate chi2-like objective functions
- Feature #677: Provide validation of GUI sample for correctness and corresponding info widget
- Feature #758: Provide recording of stack trace in crashing GUI application
- Feature #759: Integrate QuickSimulationView into JobView
- Feature #769: Remove SimuletonDataModel
- Feature #778: Windows installer: implement add/remove BornAgain desktop icon
- Feature #783: Design BornAgain main application icon
- Feature #784: Revise workspace behaviour in DesignerScene
- Feature #785: Integrate correct handling of simulation failure in JobItem
- Feature #786: Implement simple crash handler widget to report bugs
- Feature #806: Implement crash handler manager to launch external executable in platform independent stack trace retrieval
- Feature #814: Implement exceptions catching in the Core to report exception from a thread
- Feature #830: Allow DA, LMA, SSCA to ParticleLayout and propagate to GUI
- Feature #831: Implement reset of JobItem's sample and instrument models to the original.
- Feature #832: Revise submit job logic
- Feature #823: Allow multiple Layout objects per layer
- Feature #825: Update default behaviour of OutputDataView
- Feature #828: Trivial form factor for demonstration purposes
- Documentation #487: Provide screenshots for project homepage
- Documentation #781: Provide short description of GUI functionality
- Refactoring #786: Remove unnecessary calls to getOutputCoefficients
- Refactoring #919: Review SimulationParameters

---

**Backlog**

- item121
- item120
- ... 
- item95
- item93

**Sprint**

- item121
- Item95
- Item93

1 month
Agile development

Workflow is managed via Redmine management tool
- user request handling, bug tracking tool, issue tracking

http://apps.jcns.fz-juelich.de/redmine/projects/bornagain/issues

<table>
<thead>
<tr>
<th>Tracker</th>
<th>Status</th>
<th>Priority</th>
<th>Subject</th>
<th>Assignee</th>
<th>Target version</th>
</tr>
</thead>
<tbody>
<tr>
<td>829</td>
<td>Bug</td>
<td>Sprint</td>
<td>High</td>
<td>CMak...</td>
<td>pospelov</td>
</tr>
<tr>
<td>825</td>
<td>Feature</td>
<td>Sprint</td>
<td>Normal</td>
<td>Update...</td>
<td>mahacl</td>
</tr>
<tr>
<td>823</td>
<td>Feature</td>
<td>Sprint</td>
<td>Normal</td>
<td>Allow...</td>
<td>herck</td>
</tr>
<tr>
<td>811</td>
<td>Bug</td>
<td>New</td>
<td>Normal</td>
<td>FitSu...</td>
<td></td>
</tr>
<tr>
<td>791</td>
<td>Bug</td>
<td>New</td>
<td>Normal</td>
<td>LLDat...</td>
<td></td>
</tr>
<tr>
<td>788</td>
<td>Bug</td>
<td>Sprint</td>
<td>Low</td>
<td>Fix rele...</td>
<td></td>
</tr>
<tr>
<td>784</td>
<td>Feature</td>
<td>Sprint</td>
<td>Normal</td>
<td>Revise...</td>
<td></td>
</tr>
<tr>
<td>783</td>
<td>Feature</td>
<td>Sprint</td>
<td>Normal</td>
<td>Design...</td>
<td></td>
</tr>
<tr>
<td>780</td>
<td>Bug</td>
<td>Sprint</td>
<td>Normal</td>
<td>Windows...</td>
<td></td>
</tr>
<tr>
<td>776</td>
<td>Bug</td>
<td>Sprint</td>
<td>Normal</td>
<td>GUI:...</td>
<td></td>
</tr>
<tr>
<td>758</td>
<td>Feature</td>
<td>Sprint</td>
<td>Normal</td>
<td>Provide...</td>
<td></td>
</tr>
<tr>
<td>809</td>
<td>Feature</td>
<td>New</td>
<td>Normal</td>
<td>Make...</td>
<td></td>
</tr>
<tr>
<td>808</td>
<td>Feature</td>
<td>New</td>
<td>Normal</td>
<td>Compile...</td>
<td></td>
</tr>
<tr>
<td>807</td>
<td>Feature</td>
<td>Sprint</td>
<td>Normal</td>
<td>^ Make...</td>
<td></td>
</tr>
<tr>
<td>806</td>
<td>Feature</td>
<td>Sprint</td>
<td>Normal</td>
<td>^ Implement...</td>
<td></td>
</tr>
<tr>
<td>757</td>
<td>Feature</td>
<td>New</td>
<td>High</td>
<td>Implement...</td>
<td></td>
</tr>
<tr>
<td>726</td>
<td>Feature</td>
<td>Backlog</td>
<td>Normal</td>
<td>Review...</td>
<td></td>
</tr>
<tr>
<td>698</td>
<td>Feature</td>
<td>Backlog</td>
<td>Normal</td>
<td>Fix incom...</td>
<td></td>
</tr>
<tr>
<td>697</td>
<td>Feature</td>
<td>Backlog</td>
<td>Normal</td>
<td>Use Monte...</td>
<td></td>
</tr>
</tbody>
</table>
Quality assurance

Unit test control
- small test cases are executed after each compilation to ensure code stability

Nightly build system
- based on TeamCity
- automatic nightly compilation and testing
- notification about problems via e-mail

BornAgain framework

Nightly build (CMake nightly build)

<table>
<thead>
<tr>
<th>#</th>
<th>Results</th>
<th>Artifacts</th>
<th>Changes</th>
<th>Started</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>#405</td>
<td><strong>Tests passed: 243</strong></td>
<td>None</td>
<td>No changes</td>
<td>07 Oct 14 00:00</td>
<td>9m:10s</td>
</tr>
<tr>
<td>#404</td>
<td><strong>Tests passed: 243</strong></td>
<td>None</td>
<td>No changes</td>
<td>06 Oct 14 00:00</td>
<td>9m:10s</td>
</tr>
<tr>
<td>#403</td>
<td><strong>Tests passed: 243</strong></td>
<td>None</td>
<td>No changes</td>
<td>05 Oct 14 00:00</td>
<td>9m:11s</td>
</tr>
<tr>
<td>#402</td>
<td><strong>Tests passed: 243</strong></td>
<td>None</td>
<td>No changes</td>
<td>04 Oct 14 00:00</td>
<td>9m:11s</td>
</tr>
<tr>
<td>#401</td>
<td><strong>Tests passed: 243</strong></td>
<td>None</td>
<td>changes (9)</td>
<td>03 Oct 14 00:00</td>
<td>9m:10s</td>
</tr>
<tr>
<td>#400</td>
<td><strong>Tests passed: 243</strong></td>
<td>None</td>
<td>changes (9)</td>
<td>02 Oct 14 00:00</td>
<td>9m:08s</td>
</tr>
<tr>
<td>#399</td>
<td><strong>Tests passed: 243</strong></td>
<td>None</td>
<td>changes (7)</td>
<td>01 Oct 14 00:00</td>
<td>9m:06s</td>
</tr>
<tr>
<td>#398</td>
<td><strong>Tests passed: 243</strong></td>
<td>None</td>
<td>changes (14)</td>
<td>30 Sep 14 00:00</td>
<td>9m:06s</td>
</tr>
<tr>
<td>#397</td>
<td><strong>Tests passed: 243</strong></td>
<td>None</td>
<td>No changes</td>
<td>29 Sep 14 00:00</td>
<td>9m:07s</td>
</tr>
<tr>
<td>#396</td>
<td><strong>Tests passed: 243</strong></td>
<td>None</td>
<td>No changes</td>
<td>28 Sep 14 00:00</td>
<td>9m:08s</td>
</tr>
</tbody>
</table>
Quality assurance

Functional tests

- Higher level tests produce scattering plots which are automatically analyzed
- Around 80 cases nightly tested through Python and C++ API’s
  - Different scattering geometries
  - Variety of samples
  - Test of fitting engines
  - Tests of GUI

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
<th>Result</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>58/78</td>
<td>layerwithroughness.py</td>
<td>Passed</td>
<td>0.63 sec</td>
</tr>
<tr>
<td>59/78</td>
<td>ripple2_sym.py</td>
<td>Passed</td>
<td>0.21 sec</td>
</tr>
<tr>
<td>60/78</td>
<td>ripple2_asym.py</td>
<td>Passed</td>
<td>0.23 sec</td>
</tr>
<tr>
<td>61/78</td>
<td>ripple1.py</td>
<td>Passed</td>
<td>5.43 sec</td>
</tr>
<tr>
<td>62/78</td>
<td>customformfactor.py</td>
<td>Passed</td>
<td>0.38 sec</td>
</tr>
<tr>
<td>63/78</td>
<td>montecarlo_integration.py</td>
<td>Passed</td>
<td>0.68 sec</td>
</tr>
<tr>
<td>64/78</td>
<td>beam_divergence.py</td>
<td>Passed</td>
<td>0.47 sec</td>
</tr>
<tr>
<td>65/78</td>
<td>detector_resolution.py</td>
<td>Passed</td>
<td>0.20 sec</td>
</tr>
<tr>
<td>66/78</td>
<td>PyScript_isgisaxs01</td>
<td>Passed</td>
<td>0.28 sec</td>
</tr>
<tr>
<td>67/78</td>
<td>PyScript_isgisaxs04_1DDL</td>
<td>Passed</td>
<td>0.26 sec</td>
</tr>
<tr>
<td>68/78</td>
<td>PyScript_isgisaxs04_2DDL</td>
<td>Passed</td>
<td>7.50 sec</td>
</tr>
<tr>
<td>69/78</td>
<td>PyScript_isgisaxs06a</td>
<td>Passed</td>
<td>0.31 sec</td>
</tr>
<tr>
<td>70/78</td>
<td>PyScript_isgisaxs06b</td>
<td>Passed</td>
<td>0.30 sec</td>
</tr>
<tr>
<td>71/78</td>
<td>PyScript_isgisaxs07</td>
<td>Passed</td>
<td>0.29 sec</td>
</tr>
<tr>
<td>72/78</td>
<td>PyScript_isgisaxs11</td>
<td>Passed</td>
<td>0.22 sec</td>
</tr>
<tr>
<td>73/78</td>
<td>PyScript_multipartlayoutbuilder</td>
<td>Passed</td>
<td>0.30 sec</td>
</tr>
<tr>
<td>74/78</td>
<td>testfit_UsingBuilder.py</td>
<td>Passed</td>
<td>1.26 sec</td>
</tr>
<tr>
<td>75/78</td>
<td>testfit_GSLLMA.py</td>
<td>Passed</td>
<td>0.29 sec</td>
</tr>
<tr>
<td>76/78</td>
<td>testfit_GSLMultiMin.py</td>
<td>Passed</td>
<td>1.46 sec</td>
</tr>
<tr>
<td>77/78</td>
<td>testfit_GSLSimAn.py</td>
<td>Passed</td>
<td>4.46 sec</td>
</tr>
<tr>
<td>78/78</td>
<td>testfit_Minuit2.py</td>
<td>Passed</td>
<td>0.58 sec</td>
</tr>
</tbody>
</table>

100% tests passed, 0 tests failed out of 78
Total Test time (real) = 70.02 sec

BornAgain framework
Erlangen GISAXS Workshop, May 6, 2015
Status and plans

What we have right now

- Project infrastructure is settled
- Functionality often surpasses other solutions
- Software architecture is flexible and allows extension
- Small user community

Before the end 2015

- Magnetic domains and magnetic roughness in the kernel
- Fitting from GUI

Horizon 2020 Initiative

- BornAgain as a community project for GISAS and Reflectometry
- Fitting of GISAS, Off-Specular and Specular data in a single framework
BornAgain
making DWBA manageable

Thank you
Backup
Running simulation from Python

Spherical nano particles at hexagonal lattice

- Define materials
  - refractive index

- Define layers
  - materials, roughness, thicknesses

- Include particles in air layer
  - shape, size, constituting materials
  - Interference, density, position, orientation

- Define beam and detector

- Run simulation

- Save simulated detector image
Running simulation from Python

Writing a Python script with simulation description

```python
from libBornAgainCore import *

air = HomogeneousMaterial("Air", 0.0, 0.0)
substr = HomogeneousMaterial("Substr", 6e-6, 2e-8)
gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)

air_layer = Layer(air)
substrate_layer = Layer(substr)

sphere_ff = FormFactorTruncatedSphere(5*nm, 6*nm)
sphere = Particle(gold, sphere_ff)

interf = Interference2DLattice(hexagonal, 20*nm)
pdf = Distribution2DCauchy(10*nm, 10*nm)
interf.setProbabilityDistribution(pdf)

layout = ParticleLayout()
layout.addParticle(sphere)
layout.addInterferenceFunction(interference)
layout.setApproximation(LMA)

air_layer.setLayout(layout)

multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)

sim = Simulation()
sim.setDetector(100, -1*deg, 1*deg, 100, 0*deg, 2*deg)
sim.setBeam(1.0*angstrom, 0.2*deg, 0*deg)
sim.setSample(multi_layer)

sim.runSimulation()
IOFactory.writeIntensityData(sim.getIntensityData())
```
Running simulation from Python

- **Define materials**

```python
from libBornAgainCore import *

air = HomogeneousMaterial("Air", 0.0, 0.0)
substr = HomogeneousMaterial("Substr", 6e-6, 2e-8)
gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)

air_layer = Layer(air)
substrate_layer = Layer(substr)

sphere_ff = FormFactorTruncatedSphere(5*nm, 6*nm)
sphere = Particle(gold, sphere_ff)

interf = Interference2DLattice(hexagonal, 20*nm)
pdf = Distribution2DCauchy(10*nm, 10*nm)
interf.setProbabilityDistribution(pdf)

layout = ParticleLayout()
layout.addParticle(sphere)
layout.addInterferenceFunction(interference)
layout.setApproximation(LMA)

air_layer.setLayout(layout)

multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)

sim = Simulation()
sim.setDetector(100, -1*deg, 1*deg, 100, 0*deg, 2*deg)
sim.setBeam(1.0*angstrom, 0.2*deg, 0*deg)
sim.setSample(multi_layer)

sim.runSimulation()
IOFactory.writeIntensityData(sim.getIntensityData())
```
Running simulation from Python

- Define materials
  - ambience material
  - substrate material
  - Particle material

- Define layers
  - air layer
  - substrate layer

```python
from libBornAgainCore import *

air = HomogeneousMaterial("Air", 0.0, 0.0)
substr = HomogeneousMaterial("Substr", 6e-6, 2e-8)
gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)

air_layer = Layer(air)
substrate_layer = Layer(substr)

sphere_ff = FormFactorTruncatedSphere(5*nm, 6*nm)
sphere = Particle(gold, sphere_ff)

interf = Interference2DLattice(hexagonal, 20*nm)
pdf = Distribution2DCauchy(10*nm, 10*nm)
interf.setProbabilityDistribution(pdf)

layout = ParticleLayout()
layout.addParticle(sphere)
layout.addInterferenceFunction(interference)
layout.setApproximation(LMA)

air_layer.setLayout(layout)

multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)

sim = Simulation()
sim.setDetector(100, -1*deg, 1*deg, 100, 0*deg, 2*deg)
sim.setBeam(1.0*angstrom, 0.2*deg, 0*deg)
sim.setSample(multi_layer)
sim.runSimulation()
IOFactory.writeIntensityData(sim.getIntensityData())
```
Running simulation from Python

- Define materials
  - ambience material
  - substrate material
  - Particle material

- Define layers
  - air layer
  - substrate layer

- Define nano particles
  - 5 nm
  - 6 nm
  - by the material and form factor

```python
from libBornAgainCore import *

air = HomogeneousMaterial("Air", 0.0, 0.0)
substr = HomogeneousMaterial("Substr", 6e-6, 2e-8)
gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)

air_layer = Layer(air)
substrate_layer = Layer(substr)

sphere_ff = FormFactorTruncatedSphere(5\*nm, 6\*nm)
sphere = Particle(gold, sphere_ff)

interf = Interference2DLattice(hexagonal, 20\*nm)
pdf = Distribution2DCauchy(10\*nm, 10\*nm)
interf.setProbabilityDistribution(pdf)

layout = ParticleLayout()
layout.addParticle(sphere)
layout.addInterferenceFunction(interference)
layout.setApproximation(LMA)

air_layer.setLayout(layout)
multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)

sim = Simulation()
sim.setDetector(100, -1*deg, 1*deg, 100, 0*deg, 2*deg)
sim.setBeam(1.0*angstrom, 0.2*deg, 0*deg)
sim.setSample(multi_layer)

sim.runSimulation()
IOFactory.writeIntensityData(sim.getIntensityData())
```
Running simulation from Python

- Define type of interference

```python
from libBornAgainCore import *

air = HomogeneousMaterial("Air", 0, 0)
substr = HomogeneousMaterial("Substr", 6e-6, 2e-8)
gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)

air_layer = Layer(air)
substrate_layer = Layer(substr)

sphere_ff = FormFactorTruncatedSphere(5*nm, 6*nm)
sphere = Particle(gold, sphere_ff)

interf = Interference2DLattice(hexagonal, 20*nm)
pdf = Distribution2DCauchy(10*nm, 10*nm)
interf.setProbabilityDistribution(pdf)

layout = ParticleLayout()
layout.addParticle(sphere)
layout.addInterferenceFunction(interference)
layout.setApproximation(LMA)

air_layer.setLayout(layout)

multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)

sim = Simulation()
sim.setDetector(100, -1*deg, 1*deg, 100, 0*deg, 2*deg)
sim.setBeam(1.0*angstrom, 0.2*deg, 0*deg)
sim.setSample(multi_layer)

sim.runSimulation()
IOFactory.writeIntensityData(sim.getIntensityData())
```
Running simulation from Python

- Define type of interference

- Define particle layout

```python
from libBornAgainCore import *

air = HomogeneousMaterial("Air", 0.0, 0.0)
substr = HomogeneousMaterial("Substr", 6e-6, 2e-8)
gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)

air_layer = Layer(air)
substrate_layer = Layer(substr)

sphere_ff = FormFactorTruncatedSphere(5*nm, 6*nm)
sphere = Particle(gold, sphere_ff)

interf = Interference2DLattice(hexagonal, 20*nm)
pdf = Distribution2DCauchy(10*nm, 10*nm)
interf.setProbabilityDistribution(pdf)

layout = ParticleLayout()
layout.addParticle(sphere)
layout.addInterferenceFunction(interference)
layout.setApproximation(LMA)

air_layer.setLayout(layout)

multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)

sim = Simulation()
sim.setDetector(100, -1*deg, 1*deg, 100, 0*deg, 2*deg)
sim.setBeam(1.0*angstrom, 0.2*deg, 0*deg)
sim.setSample(multi_layer)

sim.runSimulation()
IOFactory.writeIntensityData(sim.getIntensityData())
```
Running simulation from Python

- Define type of interference
  ```python
  from libBornAgainCore import *
  air = HomogeneousMaterial("Air", 0.0, 0.0)
  substr = HomogeneousMaterial("Substr", 6e-6, 2e-8)
  gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)
  air_layer = Layer(air)
  substrate_layer = Layer(substr)
  sphere_ff = FormFactorTruncatedSphere(5*nm, 6*nm)
  sphere = Particle(gold, sphere_ff)
  interf = Interference2DLattice(hexagonal, 20*nm)
  pdf = Distribution2DCauchy(10*nm, 10*nm)
  interf.setProbabilityDistribution(pdf)
  layout = ParticleLayout()
  layout.addParticle(sphere)
  layout.addInterferenceFunction(interference)
  layout.setApproximation(LMA)
  air_layer.setLayout(layout)
  ```

- Define particle layout
  ```python
  multi_layer = MultiLayer()
  multi_layer.addLayer(air_layer)
  multi_layer.addLayer(substrate_layer)
  sim = Simulation()
  sim.setDetector(100, -1*deg, 1*deg, 100, 0*deg, 2*deg)
  sim.setBeam(1.0*angstrom, 0.2*deg, 0*deg)
  sim.setSample(multi_layer)
  sim.runSimulation()
  IOFactory.writeIntensityData(sim.getIntensityData())
  ```

- Assign layout to air layer
Running simulation from Python

- Assemble multi layer

```python
from libBornAgainCore import *

air = HomogeneousMaterial("Air", 0.0, 0.0)
substr = HomogeneousMaterial("Substr", 6e-6, 2e-8)
gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)

air_layer = Layer(air)
substrate_layer = Layer(substr)

sphere_ff = FormFactorTruncatedSphere(5*nm, 6*nm)
sphere = Particle(gold, sphere_ff)

interf = Interference2DLattice(hexagonal, 20*nm)
pdf = Distribution2DCauchy(10*nm, 10*nm)
interf.setProbabilityDistribution(pdf)

layout = ParticleLayout()
layout.addParticle(sphere)
layout.addInterferenceFunction(interference)
layout.setApproximation(LMA)

air_layer.setLayout(layout)

multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)

sim = Simulation()
sim.setDetector(100, -1*deg, 1*deg, 100, 0*deg, 2*deg)
sim.setBeam(1.0*angstrom, 0.2*deg, 0*deg)
sim.setSample(multi_layer)

sim.runSimulation()
IOFactory.writeIntensityData(sim.getIntensityData())
```
Running simulation from Python

- Define multi layer

- Define simulation object

```python
from libBornAgainCore import *

air = HomogeneousMaterial("Air", 0.0, 0.0)
substr = HomogeneousMaterial("Substr", 6e-6, 2e-8)
gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)

air_layer = Layer(air)
substrate_layer = Layer(substr)

sphere_ff = FormFactorTruncatedSphere(5*nm, 6*nm)
sphere = Particle(gold, sphere_ff)

interf = Interference2DLattice(hexagonal, 20*nm)
pdf = Distribution2DCauchy(10*nm, 10*nm)
interf.setProbabilityDistribution(pdf)

layout = ParticleLayout()
layout.addParticle(sphere)
layout.addInterferenceFunction(interference)
layout.setApproximation(LMA)

air_layer.setLayout(layout)

multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)

sim = Simulation()
sim.setDetector(100, -1*deg, 1*deg, 100, 0*deg, 2*deg)
sim.setBeam(1.0*angstrom, 0.2*deg, 0*deg)
sim.setSample(multi_layer)

sim.runSimulation()
IOFactory.writeIntensityData(sim.getIntensityData())
```
Running simulation from Python

- Run simulation

Simulated intensity as a function of outgoing $\alpha_f$, $\phi_f$ angles

```python
from libBornAgainCore import *

air = HomogeneousMaterial("Air", 0.0, 0.0)
substr = HomogeneousMaterial("Substr", 6e-6, 2e-8)
gold = HomogeneousMaterial("Gold", 6e-4, 2e-8)

air_layer = Layer(air)
substrate_layer = Layer(substr)

sphere_ff = FormFactorTruncatedSphere(5*nm, 6*nm)
sphere = Particle(gold, sphere_ff)

interf = Interference2DLattice(hexagonal, 20*nm)
pdf = Distribution2DCauchy(10*nm, 10*nm)
interf.setProbabilityDistribution(pdf)

layout = ParticleLayout()
layout.addParticle(sphere)
layout.addInterferenceFunction(interference)
layout.setApproximation(LMA)

air_layer.setLayout(layout)

multi_layer = MultiLayer()
multi_layer.addLayer(air_layer)
multi_layer.addLayer(substrate_layer)

sim = Simulation()
sim.setDetector(100, -1*deg, 1*deg, 100, 0*deg, 2*deg)
sim.setBeam(1.0*angstrom, 0.2*deg, 0*deg)
sim.setSample(multi_layer)

sim.runSimulation()
I0Factory.writeIntensityData(sim.getIntensityData())
```
Fitting in BornAgain

- Estimates optimal parameters in numerical model by minimizing the difference between model and reference data

User Data

- Sample description
- Experimental setup
- Fitting parameters
- Reference data
- Minimizer settings
Fitting in BornAgain

- Estimates optimal parameters in numerical model by minimizing the difference between model and reference data

User Data

- Sample description
- Experimental setup
- Fitting parameters
- Reference data
- Minimizer settings

BornAgain’s Fit Suite

- Optimal sample parameters
- Intensity plot
Fitting in BornAgain

- Estimates optimal parameters in numerical model by minimizing the difference between model and reference data

User Data

- Sample description
- Experimental setup
- Fitting parameters
- Reference data
- Minimizer settings

New sample description

New values of fit parameters

Minimization

Reference data

Chi2 value

Chi2 calculations

Simulated intensity

Simulation

Optimal sample parameters

Intensity plot
Fitting in BornAgain

Main features

- Variety of multidimensional minimization algorithms and strategies
  
<table>
<thead>
<tr>
<th>Library</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minuit2</td>
<td>Migrad, Simplex, Fumili, Scan</td>
</tr>
<tr>
<td>GSL</td>
<td>Fletcher-Reeves Conjugate Gradient Algorithm</td>
</tr>
<tr>
<td></td>
<td>BFGS Conjugate Gradient Algorithm</td>
</tr>
<tr>
<td></td>
<td>Levenberg-Marquardt Algorithm</td>
</tr>
<tr>
<td></td>
<td>Simulated Annealing Algorithm</td>
</tr>
<tr>
<td>TMVA</td>
<td>Genetic Algorithm</td>
</tr>
</tbody>
</table>

- Possibility to fit every sample parameter or their combination

- Simultaneous fit of arbitrary number of datasets
Fitting in BornAgain

**Spherical nano particles at hexagonal lattice**
- Fitting radius of spheres and lattice length

Values we are going to find during the fit

- Radius = 5 nm
- Lattice length = 10 nm

Starting values of our fit parameters

- Radius = 8 nm
- Lattice length = 8 nm
How does objective function looks like?

- $\chi^2 = F(\text{sphere radius, lattice length})$

Depending on starting values of fit parameters the minimizer will find one of two local minima.
How does objective function looks like?

- \( \chi^2 = F(\text{sphere radius, lattice length}) \)

Depending on starting values of fit parameters the minimizer will find one of two local minima.
Future plans: increasing user community

**Target groups**

- Users from GISANS community
  - Instrument scientists with large user flow
  - Heads of group from GISAS field
  - Active users of FitGISAXS, IsGISAXS programs
- Users from Reflectometry
- Users without simulation experience

**What we have to provide**

- Magnetic domains
- Magnetic roughness
- Promotion
- BornAgain publication
- Specular reflectivity
- Transmission
- Good documentation
- Modern looking web site
- Rich step-by-step tutorials
Future plans: increasing user community

Target groups

- Users from GISANS community
  - Instrument scientists with large user flow
  - Heads of group from GISAS field
  - Active users of FitGISAXS, IsGISAXS programs

- Users from Reflectometry

- Users without simulation experience

What we need

- Constructive attitude
- Feedback
- Interesting samples
- Well studied samples
- Easy-to-simulate samples
- Bug reports