

Welcome to SasView

(Pre)hackthon introduction to coding in SasView

Wojtek Potrzebowski (ESS)

Paul Butler (NIST)

Outline

- Hacktahon Goals
- Build system
- Code overview
- Writing tests
- Build system
- Model Marketplace

Goals for hackathon

- Jan 13-15
- Improving SasView5 functionality and user experience
- Making SasView5 ready for NCNR "summer" school
- "Non-functional" tasks
 - Continuous integration
 - Model Marketplace
 - Calculation engine separation
- Have some fun

SasView Architecture

SasView

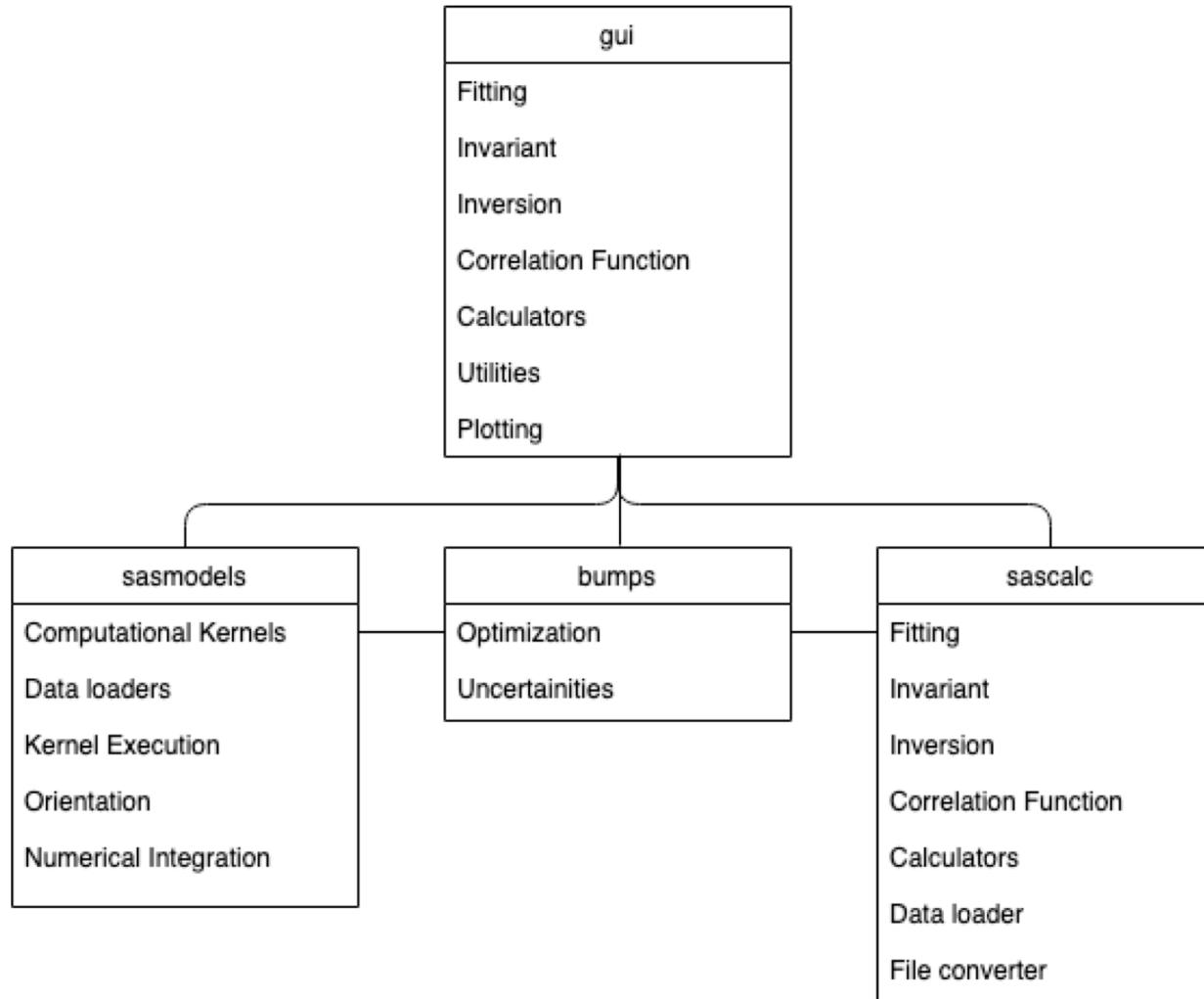
SasModels

Bumps

SasGui

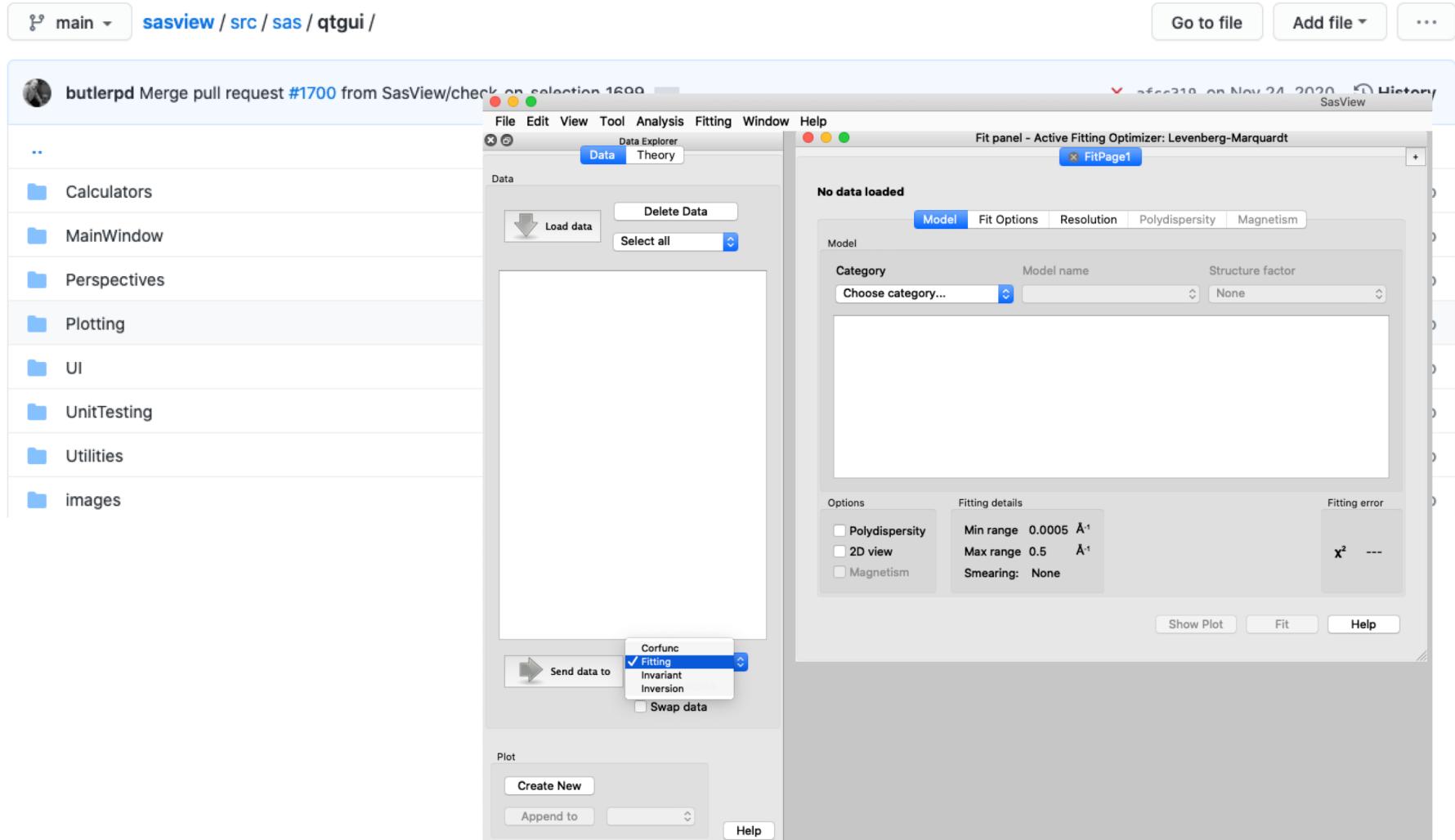
SasCalc

SasView Arhcitecture



SasGUI (qtgui)

Note: we are using **main** branch not **master**!



SasGUI Invariant Perspective

main ➔ sasview / src / sas / qtgui / Perspectives / Invariant /

krzywon Plot any corfunc and invariant plots when loading projects. Enable co... ...

..

UI	← Qt UI definitions	Addressing review findings on PR #1590 (plus new problems found)
UnitTesting	← Unit tests	Fix failing unit tests
media	← Images Doc files	Tweaks and cleanups in 5.x invariant_help
InvariantDetails.py		Addressing review findings on PR #1590 (plus new problems found)
InvariantPerspective.py		Plot any corfunc and invariant plots when loading projects. Enable co...
InvariantUtils.py		Replace all(?) references to filename when display name is more appro...
_init__.py		Initial commit of the main window prototype

Slicer code

main	sasview / src / sas / qtgui / Plotting /	Go to file	Add file	...
 murphyryanp	Merge pull request #1567 from SasView/ESS_GUI_1554_slicer	80bcf85 on Jun 2, 2020		History
..				
 Masks	Fix for 2D masking - SASVIEW-1361			14 months ago
 Slicers	Updated AnnulusSlicer to improve slicer min limits (low-q 2D data) an...			8 months ago
 UI	 MaskEditor.py Replace Apply button menu driven functionality with additional button.			2 years ago
 UnitTe	 PlotHelper.py cherry picking sascalc changes from master SASVIEW-996			2 years ago
 AddTe	 PlotProperties.py Replace Apply button menu driven functionality with additional button.			2 years ago
 Arrow:	 PlotUtilities.py Improved line colour handling for 1D data charts.			2 years ago
 Binder	 PlotTables.py Fixed erroneous error bars. SASVIEW-994			2 years ago
 BoxSu	 Plotter.py Added context menu item for legend toggle, for quick and regular 1D			8 months ago
	 Plotter2D.py ← Merge pull request #1567 from SasView/ESS_GUI_1554_slicer			7 months ago
	 PlotterBase.py Added/modified unit tests			8 months ago
	 PlotterData.py Added plot roles to Data1D/Data2D structures to allow for smoother pl...			2 years ago
	 ScaleProperties.py Replace Apply button menu driven functionality with additional button.			2 years ago
	 SetGraphRange.py Replace Apply button menu driven functionality with additional button.			2 years ago
	 SlicerModel.py ← Updating annulus slicer, batch slicing, and slicer parameters			8 months ago
	 SlicerParameters.py ← Removed the batch slicing tab and associated widgets			8 months ago
	 WindowTitle.py Replace Apply button menu driven functionality with additional button.			2 years ago
	 __init__.py cherry picking sascalc changes from master SASVIEW-996			2 years ago
	rangeSlider.py Remove copyright header from docstring			13 months ago

SasCalc

- Back-end calculations for sasgui perspective
- Data fitting:
 - src/sas/sascalc/fit/
- Pr inversion
 - src/sas/sascalc/pr/
- Data loader: src/sas/sascalc/dataloader/loader.py

SasCalc

```
from sas.sascalc.dataloader.loader import Loader
from sas.sascalc.pr.invertor import Invertor

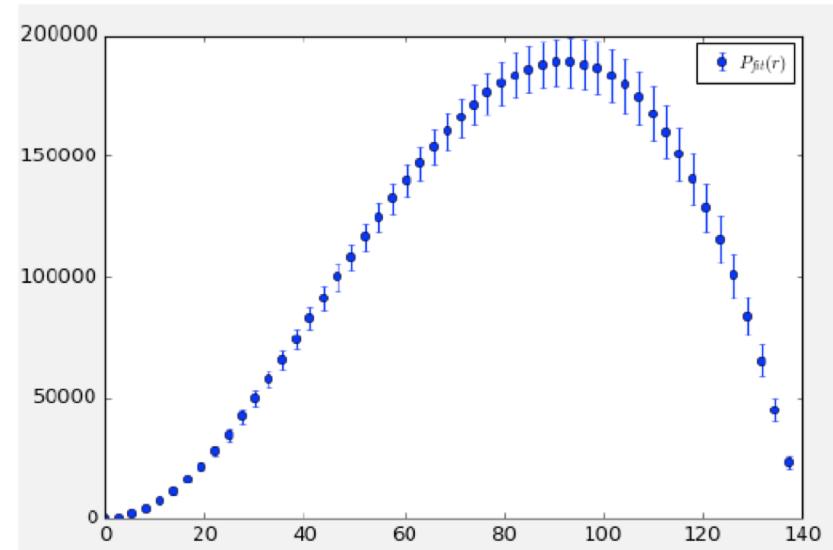
loader = Loader()
test_data = loader.load("sphere_80.txt")

pr = Invertor()

# Set data
pr.x  = test_data.x
pr.y  = test_data.y

# Perform inversion and show graph
x, y = pr.invert()

import matplotlib.pyplot as plt
plt.plot(x, y)
plt.show()
```



And more at: https://github.com/SasView/documents/blob/master/Meetings/User_Meeting_2018/SasView%2BCLI%2Boverview.html

SasModels

- Models located in: `sasmodels/models`
- Three different types of models:
 - Python only: `broad_peak.py`
 - Python with embedded C: `stickyhardsphere.py`
 - Python and separate C: `barbell.py` and `barbell.c`

SasModels essential components

- Model Description: Docstring
- Name, title, short description
- Parameters table
- C files to be included during compilation
- Unit tests

SasModels functions

- `Iq` - 1D scattering intensity functions for the case where the scatterer is randomly oriented
- `Iqxy` - the 2D scattering intensity functions provide $I(Q, \phi)$ for an oriented system as a function of a Q
- `Form_volume` - calculates the volume of particle V , which is used to normalize form factor.
- `ER` – calculates effective radius
- `VR` - calculate particle volume/total volume for shape models
- `INVALID(v)` returns `False` if `v.parameter` is invalid for some parameter or other (e.g., `v.bell_radius < v.radius`).

SasModels core modules

- Model computation:
 - `sasmodels/sasview_model.py` – interface to sasview
 - `sasmodels/kernel.py` -interface to all kernel models
 - `sasmodels/kerneldll.py` –dll driver for c kernels
 - `sasmodels/kernelpy.py` – python driver
 - `sasmodels/kernelcl.py` – opencl driver
 - `sasmodels/core.py` - prepares the model for the execution
 - `sasmodels/generate.py` – generates model file based on the template

Running bumps with sasmodels

- Sasmodels example contains data set for oriented rod-like shape
- sasmodels/examples/fit.py (sasmodels interface to bumps)
- Running
 - bumps fit.py cylinder –preview
 - Fit.py accepts two arguments: type of model and view (radial and tnagential)

Custom settings

- Local configuration
 - `~/.sasview/config/custom_config.py`
- Compiled sasmodesl
 - `~/.sasview/compiled_models/`
- Plugin models:
 - `~/.sasview/plugin_models/`
- Model categories:
 - `~/.sasview/categories.json`

Unit, GUI, sasmodels tests

- For sasview core, tests are located in sasview/test.
 - Tests are stored in subdirectories of the test directory, as *package/test/utest_*.py*.
 - Other files in the test subdirectory are data, expected output and other support files.
 - Before running any tests you must first run *setup.py build* in the root.
 - Tests are run against the current source directory with run magic to find the compiled code, so you only need to rebuild between tests if you are modifying the C code.
- For GUI, tests are located in respective folders e.g for Fitting sasview/src/sas/qtgui/Perspectives/Fitting/UnitTesting/.
 - For this part we use unit test with MagickMock (for general introduction please refer to <https://www.thedigitalcatonline.com/blog/2016/03/06/python-mocks-a-gentle-introduction-part-1/> and <https://aaronlevvier.github.io/python-unit-testing-with-magicmock/>)
- For sasmodels tests are part of model definition (see <https://github.com/SasView/sasmodels/blob/master/doc/guide/plugin.rst#test-your-new-model>)

More about tests: https://github.com/SasView/sasview/wiki/DevNotes_DevGuide_CodeTesting

Model marketplace

The screenshot shows the SasView Marketplace homepage. At the top, there is a dark header bar with the "SasView Marketplace" logo, a search input field, and a "Log In" button. The main content area has a light gray background. On the left, a large heading says "Welcome to the SasView Marketplace!". Below it, a paragraph explains the purpose of the Marketplace. To the right, a sidebar titled "Categories:" lists various model types: Cylinder, Ellipsoid, Lamellae, Other, Paracrystal, Parallelepiped, Shape-Independent, Sphere, Structure Factor, and All Models.

SasView Marketplace

Search

Log In

Welcome to the SasView Marketplace!

The Marketplace allows members of the SAS Community to contribute plug-in fitting models for the popular [SasView](#) data analysis program for all to use. (Please note: these plug-in models require [SasView version 4.0 or later](#)).

Contributed models should be written in Python (compatibility with python 2.7.x is currently required) or, if computational speed is an issue, in a combination of Python and C. You only need to upload the .py/.c source code files to the Marketplace!

Instructions on how to code a new plug-in model are available [here](#). It may also be helpful to examine the library models in the \sasmmodels-data\models sub-folder of your SasView installation directory.

You also have the option to upload a SasView text file of the scattering function data computed by your model. If you do this a graph of the scattering function will appear under the Marketplace entry for your model.

Disclaimer: Models are expected to be contributed to the Marketplace in good faith and with the best intentions, and come without any warranty as to their correctness and suitability for purpose. Neither the author of the model or the SasView Development Team will be held liable for any loss or damage arising from conclusions drawn from the use of a model contributed to the Marketplace. This includes models that are marked as 'verified' by the SasView Development Team.

Categories:

- Cylinder
- Ellipsoid
- Lamellae
- Other
- Paracrystal
- Parallelepiped
- Shape-Independent
- Sphere
- Structure Factor
- All Models

It is **python 2.7...**

Build system

https://jenkins.esss.dk/sasview-beta/

Most Visited Getting Started Intranet Chess ESS Public Web ESSnow Service Port... D+ | Uri Raviv Rese... Nyårsmedley Smygehuk...

Jenkins

search log in

People

Build History

Project Relationship

Check File Fingerprint

Convert To Pipeline

Build Queue

No builds in the queue.

Build Executor Status

master

1 Idle
2 Idle

Centos73_11.52

1 Idle

OSX1013_11.56

1 Idle

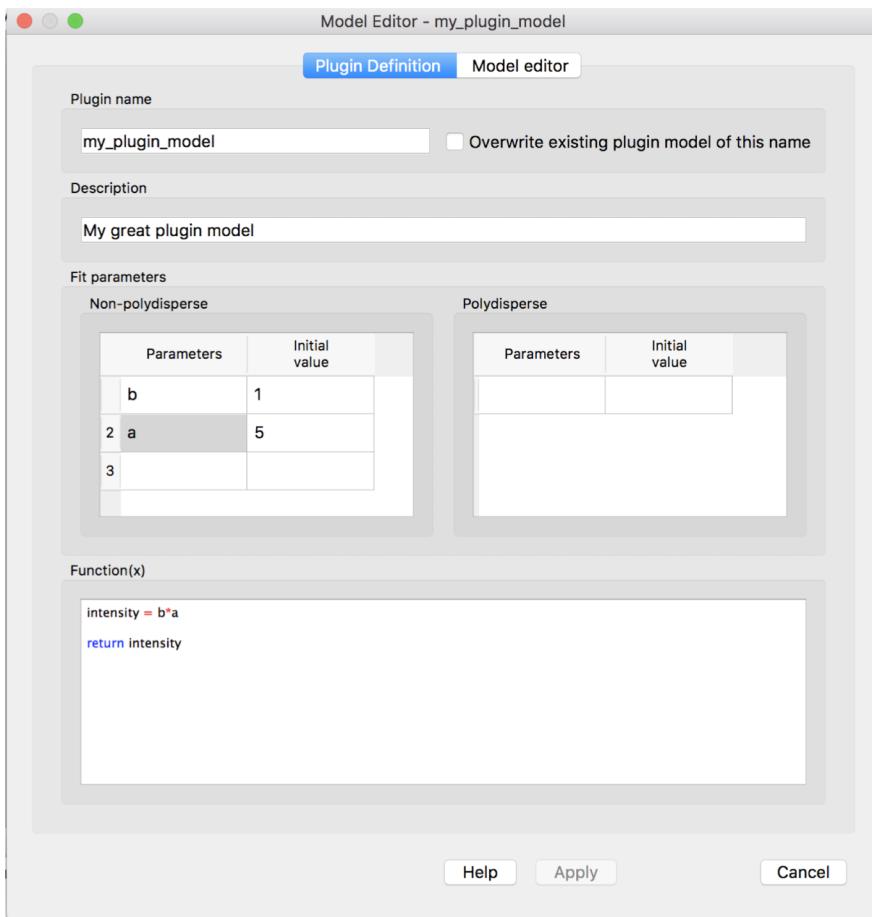
OSX1013_11.57

1 Idle

All					
S	W	Name ↓	Last Success	Last Failure	Last Duration
		sasmodes OS10.13-ghprb	4 mo 25 days - #18	2 mo 1 day - #19	10 min
		sasmodes Windows10-ghprb	2 mo 1 day - #11	5 mo 2 days - #9	34 min
		SasView-Centos	10 mo - #700	10 mo - #703	23 min
		SasView-OSX10.13-ghprb	3 mo 20 days - #454	N/A	10 min
		SasView-OSX10.13-test	1 yr 9 mo - #21	1 yr 9 mo - #65	1 min 6 sec
		SasView-OSX1013	5 days 23 hr - #919	N/A	12 min
		SasView-OSX1013-Release	11 mo - #33	11 mo - #32	20 min
		SasView-OSX1014-Release	5 mo 21 days - #94	6 mo 9 days - #90	8 min 29 sec
		SasView-Ubuntu	1 yr 12 mo - #455	1 yr 10 mo - #501	18 min
		SasView-Ubuntu1804	5 days 23 hr - #737	N/A	10 min
		SasView-Ubuntu1804-ghprb	2 mo 20 days - #13	N/A	9 min 45 sec
		SasView-Ubuntu1804-Release	5 mo 21 days - #19	N/A	9 min 2 sec
		SasView-Windows10	5 days 23 hr - #504	N/A	26 min

Becoming more and more difficult to maintain -> Github actions!

Plugin model editor (Fitting->Add Custom Model)



The screenshot shows the 'Model Editor - my_plugin_model' window with the 'Model editor' tab selected. The 'Model' section displays the generated Python code:

```
Calculates my_plugin_model.

My great plugin model

References
-----
Authorship and Verification
-----
**Author:** ---- **Date:** 2019YY-03m-19d
**Last Modified by:** ---- **Date:** 2019YY-03m-19d
**Last Reviewed by:** ---- **Date:** 2019YY-03m-19d

from math import *
from numpy import inf

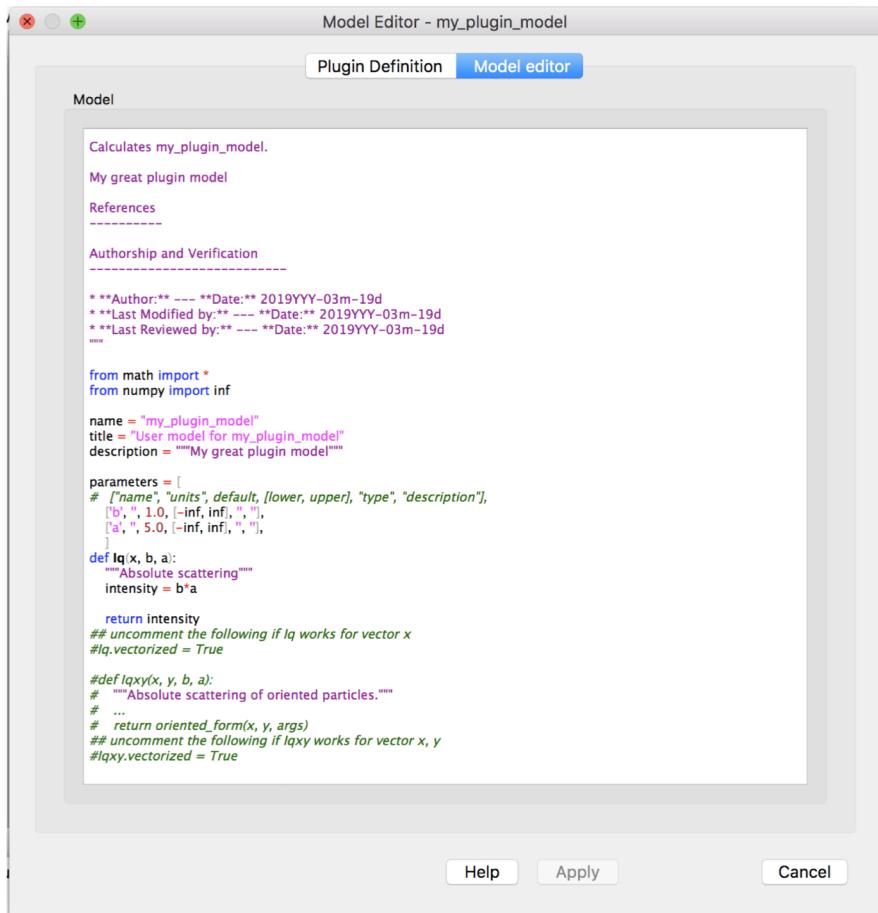
name = "my_plugin_model"
title = "User model for my_plugin_model"
description = "My great plugin model"

parameters = [
    {"name": "b", "units": "", "default": [1.0, [-inf, inf], "b", "b"], "type": "float", "description": "b"}, {"name": "a", "units": "", "default": [5.0, [-inf, inf], "a", "a"], "type": "float", "description": "a"}]
def Iq(x, b, a):
    """Absolute scattering"""
    intensity = b*a
    return intensity
## uncomment the following if Iq works for vector x
#Iq.vectorized = True

#def Iqxy(x, y, b, a):
#    """Absolute scattering of oriented particles."""
#    ...
#    # return oriented_form(x, y, args)
## uncomment the following if Iqxy works for vector x, y
#Iqxy.vectorized = True
```

At the bottom are 'Help', 'Apply', and 'Cancel' buttons.

Python and C files



The screenshot shows the 'cylinder.c' file. The code is a C implementation of scattering calculations. It includes a macro for invalid input, static functions for calculating form volume, scattering length, and oriented average form factor, and a function for calculating the total scattering intensity. The code uses M_PI and various trigonometric functions from the standard library.

```
#define INVALID(v) (v.radius<0 || v.length<0)  
  
static double  
form_volume(double radius, double length)  
{  
    return M_PI*radius*radius*length;  
}  
  
static double  
fq(double qab, double qc, double radius, double length)  
{  
    return sas_2J1x_x(qab*radius) * sas_sinx_x(qc*0.5*length);  
}  
  
static double  
orient_avg_1D(double q, double radius, double length)  
{  
    // translate a point in [-1,1] to a point in [0, pi/2]  
    const double zm = M_PI_4;  
    const double zb = M_PI_4;  
  
    double total = 0.0;  
    for (int i=0; i<GAUSS_N ; i++) {  
        const double theta = GAUSS_Z[i]*zm + zb;  
        double sin_theta, cos_theta; // slots to hold sincos function output  
        // theta (theta,phi) the projection of the cylinder on the detector plane  
        SINCOS(theta , sin_theta, cos_theta);  
        const double form = fq(q*sin_theta, q*cos_theta, radius, length);  
        total += GAUSS_W[i] * form * form * sin_theta;  
    }  
    // translate dx in [-1,1] to dx in [lower,upper]  
    return total*zm;  
}  
  
static double  
Iq(double q,  
    double sld,  
    double solvent_sld,  
    double radius,  
    double length)  
{  
    const double s = (sld - solvent_sld) * form_volume(radius, length);  
    return 1.0e-4 * s * s * orient_avg_1D(q, radius, length);  
}
```

Category manager (View->Category Manager)

