



EUROPEAN  
SPALLATION  
SOURCE



# SasView

Data Analysis for Small Angle Scattering



Science & Technology Facilities Council



# A little history ...

2006  
2011



Heritage: NIST IGOR macros  
SansView is DANSE project output  
~ 8.5% of funds were for SANS

2012  
2013

NIST Supported initial transition from NSF funding

2013

Transition to Community project.

2014

Move to GitHub  
Rename to SasView

v3.0 released  
v3.1 released

2016

v4.0 released  
v4.1 released

2017



2018

v4.2 released  
v5.0b1 released  
v4.2.1 released  
v5.0b2 released

2019

1<sup>st</sup> Code Camp at NIST April 2013

2<sup>nd</sup> Code Camp at ISIS April 2014

3<sup>rd</sup> Code Camp at ESS Feb 2015

4<sup>th</sup> Code Camp at TU Delft March 2016

5<sup>th</sup> Code Camp at ORNL Oct 2016

6<sup>th</sup> Code Camp at ILL/ESRF April 2017

7<sup>th</sup> Code Camp at DMSC October 2017

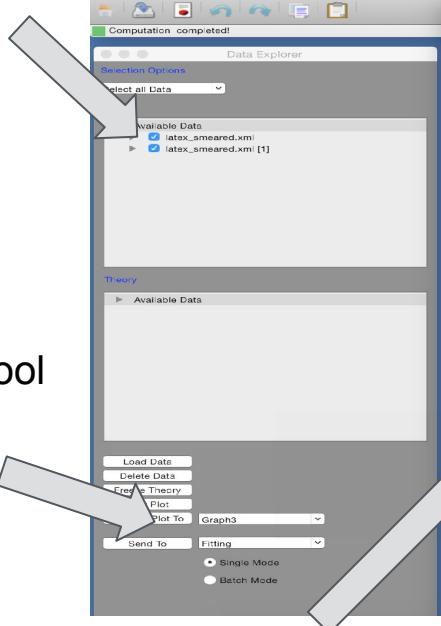
8<sup>th</sup> Code Camp at ESS Sept 2018

1<sup>st</sup> SasView User Meeting at SAS2018

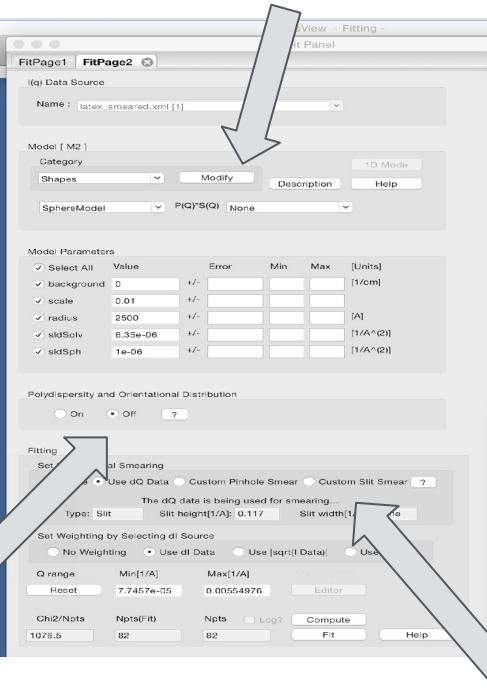
9<sup>th</sup> Code Camp at ILL/ESRF March 2019

# 1D Analysis

Data management  
Common data formats supported, including NXCansas & *cansas1D*



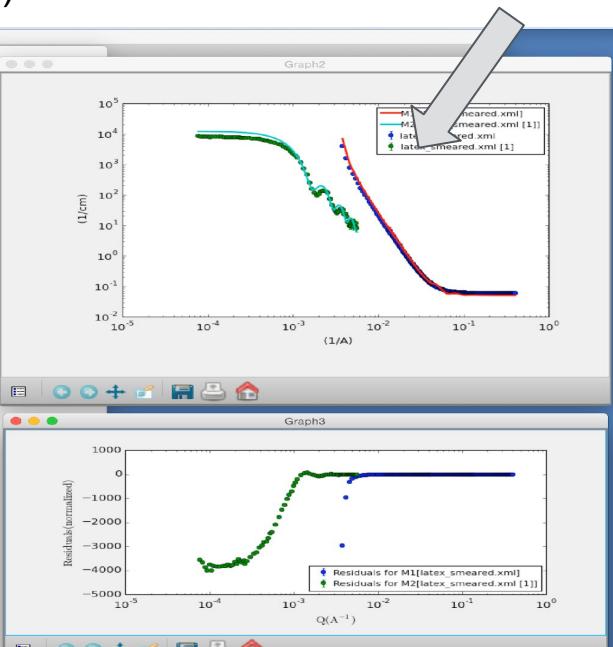
Wide choice of built-in models (> 70)  
 $P(Q)$ ,  $S(Q)$  &  $P(Q)*S(Q)$



Analysis Tool  
Choice  
&  
Plotting

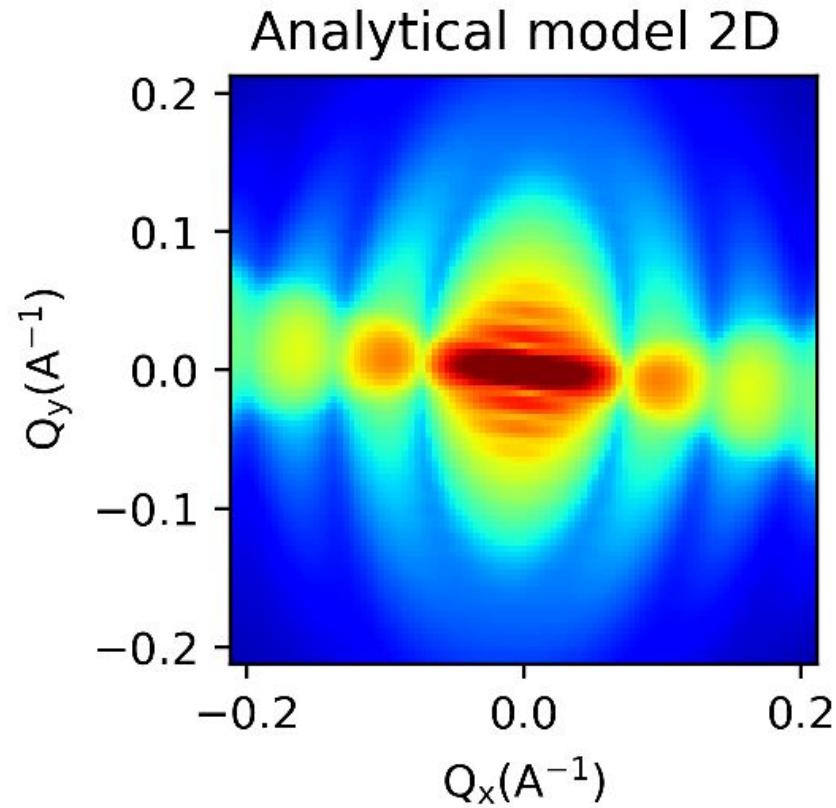
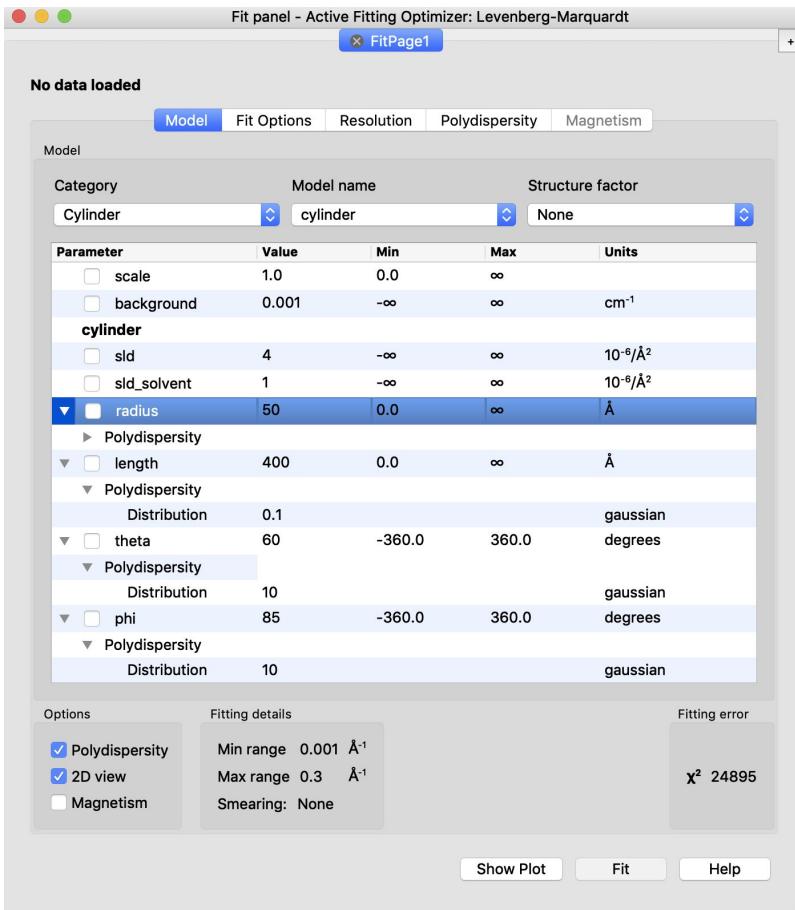
Generic parameter polydispersity  
Choice of distribution and distribution parameters

Simultaneous fitting

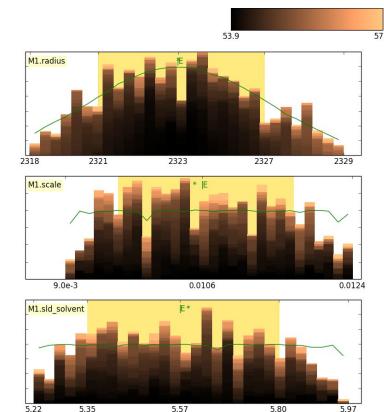
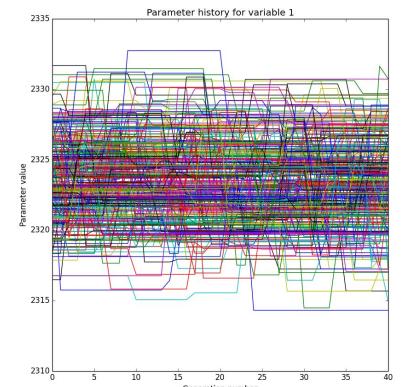
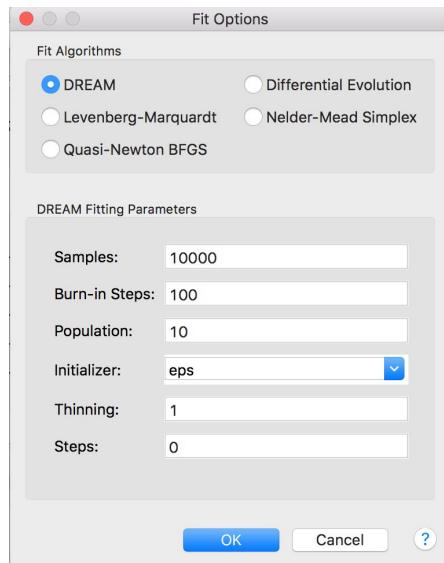
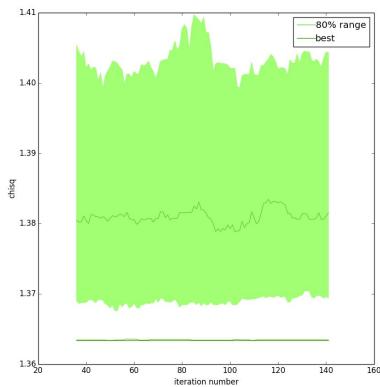
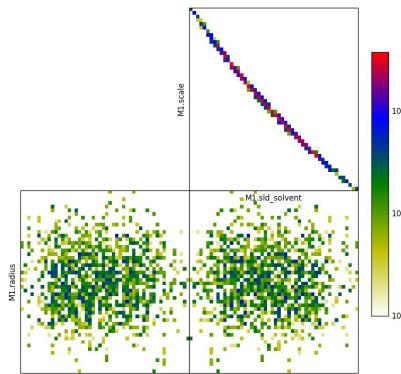


Resolution smearing (pinhole and slit)  
Automatically from data or provide parameters

# 2D Analysis



# Fitting Algorithms



Uses bumps package from P. Kienzle

# Plugin Model Editor

Model Editor - my\_plugin\_model

Plugin Definition Model editor

Plugin name: my\_plugin\_model  Overwrite existing plugin model of this name

Description: My great plugin model

Fit parameters:

Non-polydisperse

Parameters	Initial value
b	1
a	5
3	

Polydisperse

Parameters	Initial value

Function(x)

```
intensity = b*a
return intensity
```

Help Apply Cancel

Model Editor - my\_plugin\_model

Plugin Definition Model editor

Model

Calculates my\_plugin\_model.  
My great plugin model  
References  
Authorship and Verification

```
'''Author''' ---- ''Date:''' 2019YYYY-03m-19d
'''Last Modified by''' ---- ''Date:''' 2019YYYY-03m-19d
'''Last Reviewed by''' ---- ''Date:''' 2019YYYY-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", "units", default, lower, upper), "type", "description"),
    ('b', '1.0', -inf, inf, 'float', "Parameter b"),
    ('a', '5.0', -inf, inf, 'float', "Parameter a")
]

def Iq(x, b, a):
    """Absolute scattering"""
    intensity = b*a

    return intensity
    # uncomment the following if Iq works for vector x
    # Iq.vezorized = 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.vezorized = True
```

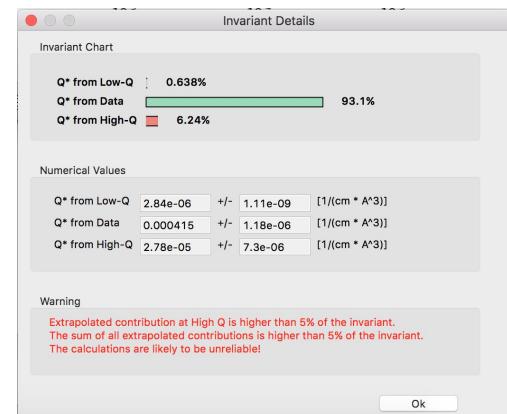
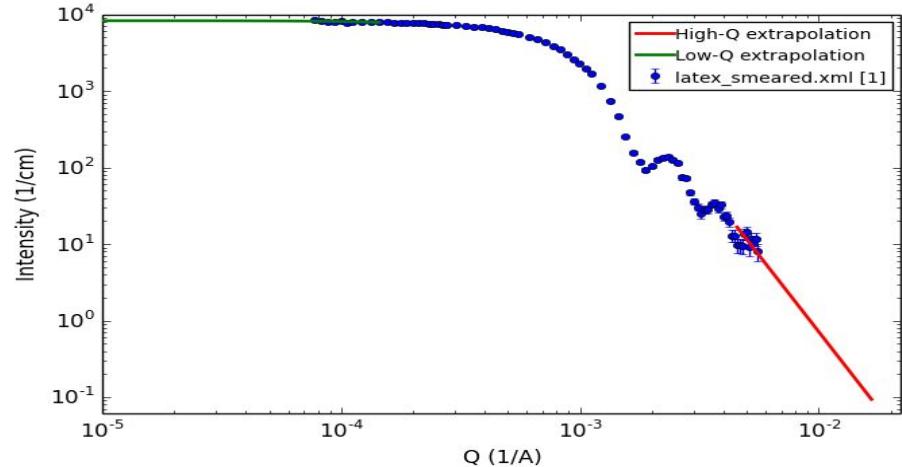
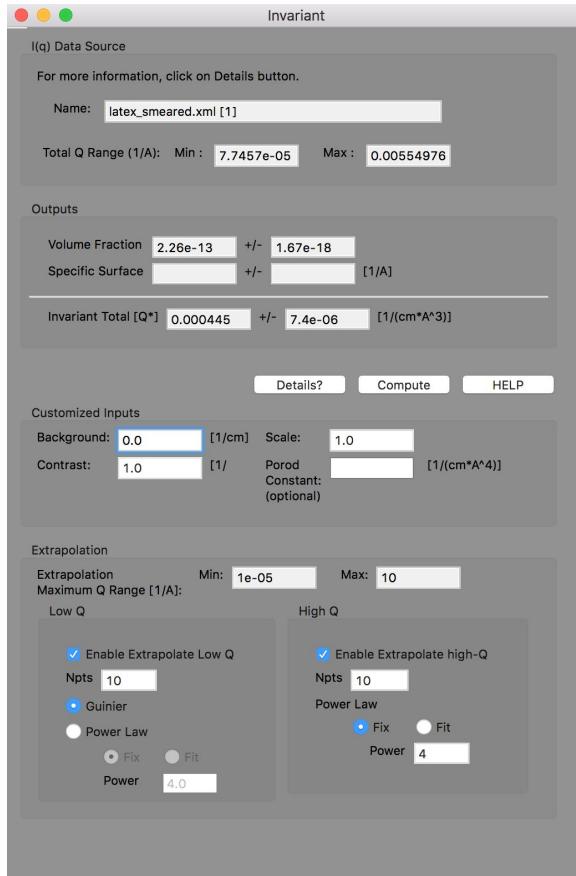
Help Apply Cancel

# Python & C model files

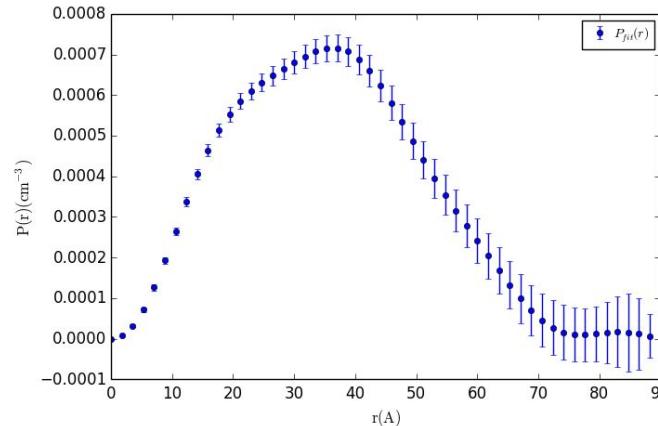
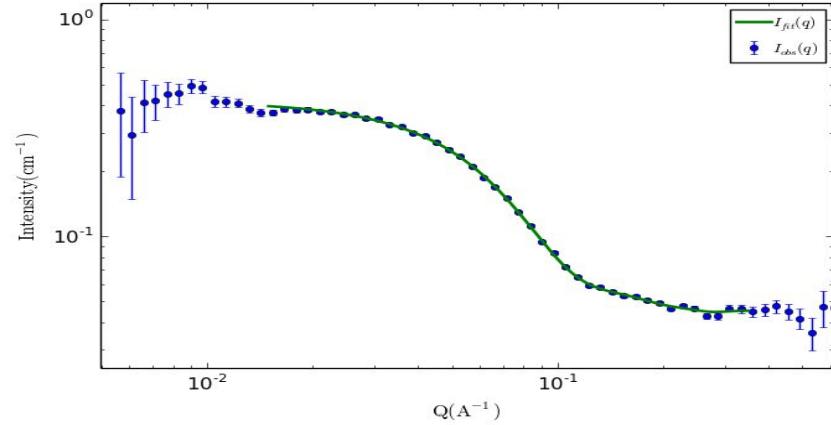
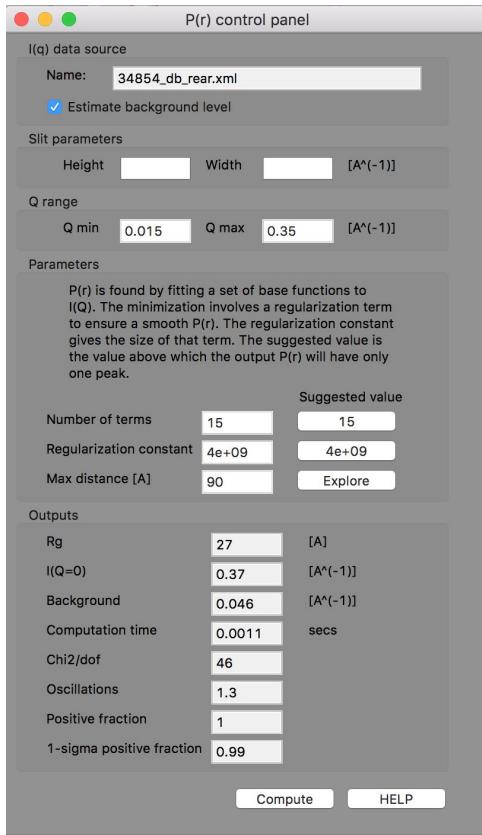
```
cylinder.py
182 import ...
184
185     name = "cylinder"
186     title = "Right circular cylinder with uniform scattering length density."
187     description = """
188         f(q, alpha) = 2*(sld - sld_solvent)*V*sin(qLcos(alpha)/2)
189         /[qLcos(alpha)/2]*J1(qRsin(alpha))/[qRsin(alpha)]
190
191         P(q, alpha) = scale/V*f(q, alpha)^2+background
192         V: Volume of the cylinder
193         R: Radius of the cylinder
194         L: Length of the cylinder
195         J1: The bessel function
196         alpha: angle between the axis of the
197             cylinder and the q-vector for 1D
198         :the output is P(q)=scale/V*integral
199         from pi/2 to zero of...
200         f(q, alpha)^2*sin(alpha)*dalpha + background
201
202     """
203
204     category = "shape:cylinder"
205
206     #         [ "name", "units", default, [lower, upper], "type", "description"],
207     parameters = [[["sld", "le-6/Ång^2", 4, [-inf, inf], "sld",
208         "Cylinder scattering length density"],
209         ["sld_solvent", "le-6/Ång^2", 1, [-inf, inf], "sld",
210         "Solvent scattering length density"],
211         ["radius", "Ång", 20, [0, inf], "volume",
212         "Cylinder radius"],
213         ["length", "Ång", 400, [0, inf], "volume",
214         "Cylinder length"],
215         ["theta", "degrees", 60, [-360, 360], "orientation",
216         "cylinder axis to beam angle"],
217         ["phi", "degrees", 60, [-360, 360], "orientation",
218         "rotation about beam"]],
219     ]
220
221
222     source = ["lib/polevl.c", "lib/sas_J1.c", "lib/gauss76.c", "cylinder.c"]
223
224     def ER(radius, length):
225         """
226             Return equivalent radius (ER)
227         """
228         ddd = 0.75 * radius * (2 * radius * length + (length + radius) * (length + pi * radius))
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
```

```
cylinder.c
1 #define INVALID(v) (v.radius<0 || v.length<0)
2
3 static double
4 form_volume(double radius, double length)
5 {
6     return M_PI*radius*radius*length;
7 }
8
9 static double
10 fq(double qab, double qc, double radius, double length)
11 {
12     return sas_2J1x_x(qab*radius) * sas_sinx_x(qc*0.5*length);
13 }
14
15 static double
16 orient_avg_1D(double q, double radius, double length)
17 {
18     // translate a point in [-1,1] to a point in [θ, π/2]
19     const double zm = M_PI_4;
20     const double zb = M_PI_4;
21
22     double total = 0.0;
23     for (int i=0; i<GAUSS_N ;i++) {
24         const double theta = GAUSS_Z[i]*zm + zb;
25         double sin_theta, cos_theta; // slots to hold sincos function output
26         // theta (theta,phi) the projection of the cylinder on the detector plane
27         SINCO5(theta , sin_theta, cos_theta);
28         const double form = fq(qsin_theta, q*cos_theta, radius, length);
29         total += GAUSS_W[i] * form * form * sin_theta;
30     }
31     // translate dx in [-1,1] to dx in [lower,upper]
32     return total*zm;
33 }
34
35 static double
36 Iq(double q,
37      double sld,
38      double solvent_sld,
39      double radius,
40      double length)
41 {
42     const double s = (sld - solvent_sld) * form_volume(radius, length);
43     return 1.0e-4 * s * s * orient_avg_1D(q, radius, length);
44 }
45
```

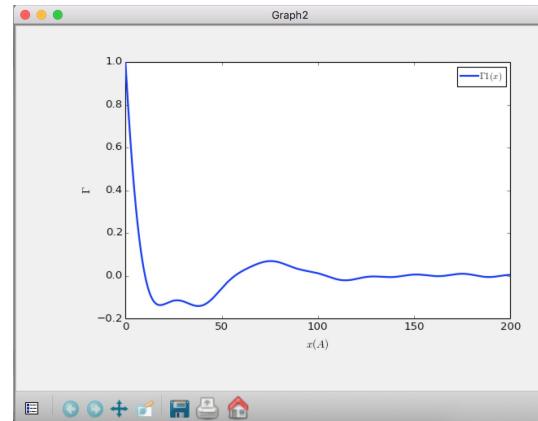
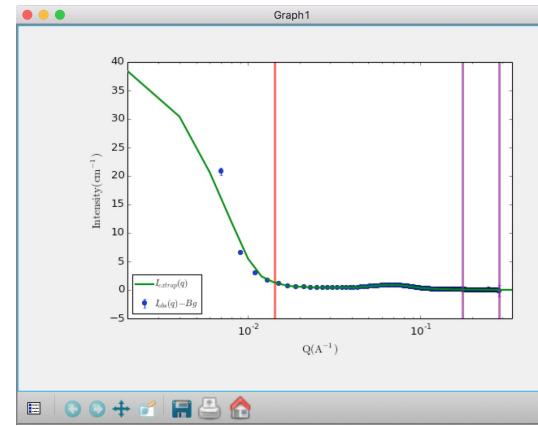
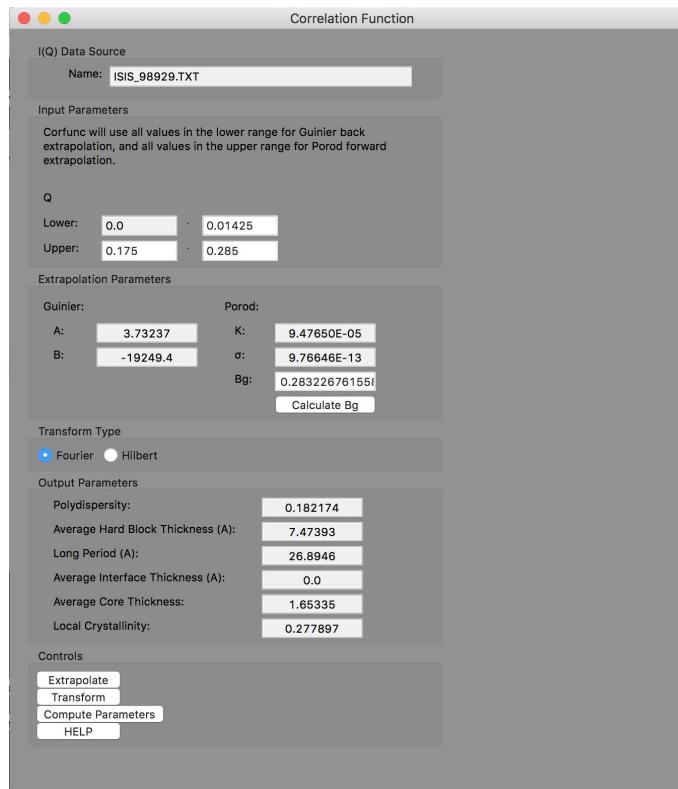
# Invariant Calculation



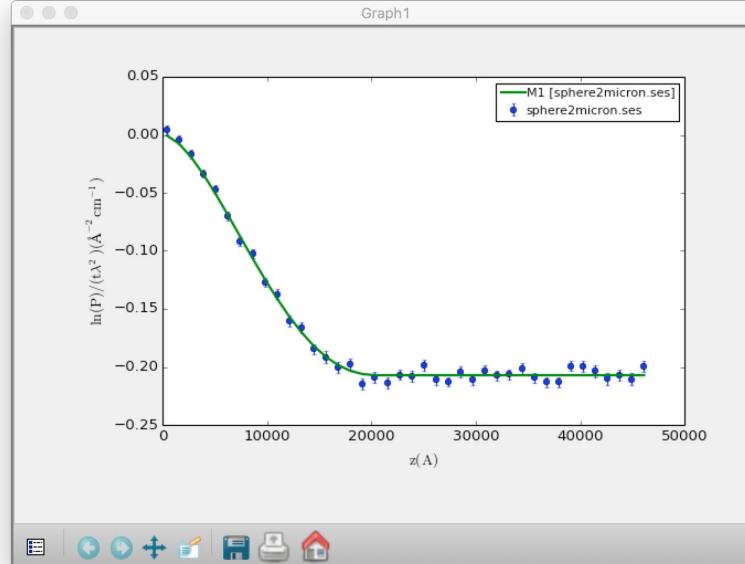
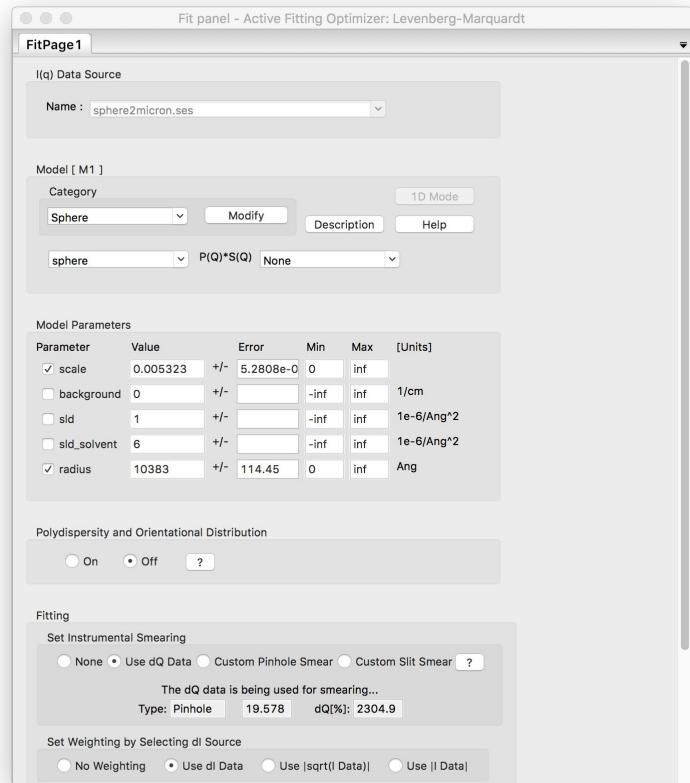
# P(r) Inversion



# Correlation Function Analysis (new!)



# SESANS Analysis



Automatic Hankel Transform of SANS models  
(TU Delft & ISIS)

# How We Work

## Open, Collaborative, Community Development

Code is open source and publicly hosted at Github

Released under BSD 3-clause license

Bug and Enhancement Ticket System

Bi-weekly developer calls

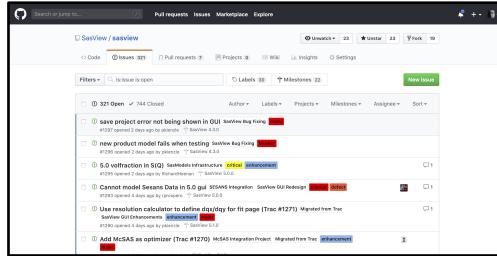
Code Camps

5 Year Roadmap

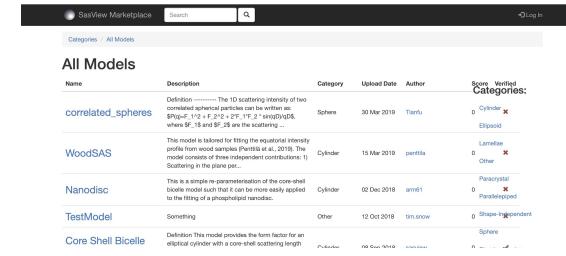
Model Marketplace

DOI for each release

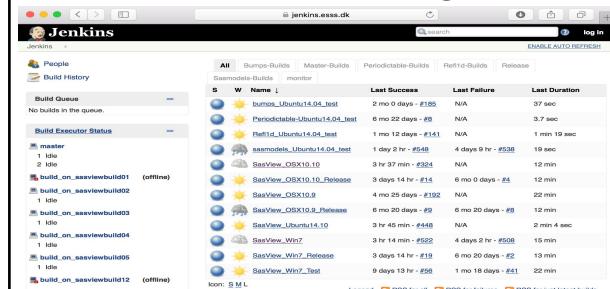
Code Hosting, Task and bug tracking, and developer/user wiki  
Github



Model Marketplace for users to share their models  
marketplace.sasview.org



Automated Builds  
build.sasview.org



# How We Work

## Open, Collaborative, Community Development

### DOI for each release

<https://zenodo.org/communities/sasview-analysis>

The screenshot shows the Zenodo interface for the SasView community. At the top, there's a search bar, an upload button, and a 'Communities' link. On the right, there's a dropdown for user authentication. Below the header, the title 'SasView - Data Analysis for Small Angle Scattering' is displayed. A 'Recent uploads' section follows, showing a list of releases. The first release is 'SasView version 4.2' (September 9, 2018), which includes a detailed description of its features and bug fixes. The second release is 'SasView version 4.1.2' (August 12, 2017), which includes a note about bug fixes and compatibility. Both releases have 'Software' and 'Open Access' buttons.

September 9, 2018

Software Open Access

### SasView version 4.2

Doucet, Mathieu; Cho, Jae Hie; Alina, Gervaise; Bakker, Jurrian; Bouwman, Wim; Butler, Paul; Campbell, Kieran; Gonzales, Miguel; Heenan, Richard; Jackson, Andrew; Juhas, Pavol; King, Stephen; Kenzle, Paul; Krzywon, Jeff; Markvardsen, Anders; Nielsen, Torben; O'Driscoll, Lewis; Potrzebowski, Wojciech; Ferraz Leal, Ricardo; Richter, Tobias; Rozyczo, Piotr; Snow, Tim; Washington, Adam

New in Version 4.2.0

This release heralds many improvements and a host of bug fixes, along with some significant changes from previous versions. Further, as promised, it marks the end of support for 32 bit operating systems and is only available for 64 bit operating systems. Wi

With this version the change to the new model API and plugins infrastructure begun with 4.0 is essentially complete (though extensions are in the works, and more are likely, they should remain backwardly compatible with previous versions of SasView).

Old-style plugin models, including old sumimultiply models, continue to be supported (i.e. SasView will run them) in 4.x, although our automatic on-the-fly translation may not cope in all use cases (see Known Issues below). However, this backward compatibility will be removed in 5.0 and users are therefore strongly encouraged to convert their custom models to the new API.

Finally, the changes to orientation angles and orientational distribution definitions are now also complete.

Changes

\* The infrastructure for calculating 2D patterns from 3D orientated objects has been totally re-factored. It is now more accurate and consistent across models.

\* The way that SasView defines the orientation of anisometric and aligned objects has been completely overhauled. It now differs from previous versions.

\* Plugin models, including sumimultiply models, have completely migrated to the new infrastructure. NOTE that 3.x type models as well as early, intermediate 4.x type models, including those generated by sumimultiply will continue to be supported in 4.x but will likely no longer be supported after the move to 5.0. Users are strongly encouraged to migrate any custom models.

<http://www.sasview.org>

<http://github.com/SasView>

# How We Work

## Open, Collaborative, Community Development

DOI for each release

<https://zenodo.org/communities/sasview-analysis>

**Publication date:**  
September 9, 2018

**DOI:**  
[DOI 10.5281/zenodo.1412041](https://doi.org/10.5281/zenodo.1412041)

**Grants:**  
[European Commission](#):  
• SINE2020 - World class Science and Innovation with Neutrons in Europe 2020 – SINE2020 (654000)

**Alternate identifiers:**  
<https://github.com/SasView/sasview/releases/tag/v4.1.2>

**Communities:**  
[SasView - Data Analysis for Small Angle Scattering](#)

**License (for files):**  
 BSD 3-Clause "New" or "Revised" License

Files (405.4 MB)	
Name	Size
SasView-4.2.0-MacOSX.dmg	196.9 MB
md5:b0c68251297d8ae56c90354af5b206a9 <small>?</small>	<a href="#"></a> Download
sasview-4.2.0.tar	66.0 MB
md5:de9803768bf49f2f5d61e671ef33b1c6 <small>?</small>	<a href="#"></a> Download
sasview-4.2.0.zip	42.4 MB
md5:590672685fd4bb7840f0dfaad0b61d54 <small>?</small>	<a href="#"></a> Preview <a href="#"></a> Download
setupSasView-4.2.0-x64.exe	100.0 MB
md5:3b10af1557127f6eb31f427b9549f6d1 <small>?</small>	<a href="#"></a> Download

<http://www.sasview.org>

<http://github.com/SasView>

# How We Work

## Open, Collaborative, Community Development

We work together towards common goals formulated through community input, with two guiding principles ...

*He who pays the piper ...*

Those who bring the resources (time and effort, or funds to buy time and effort) choose what to work on.

*You break it, you bought it ...*

You are not allowed to break what is already there for others. If you break it, you fix it.

# How We Work

## Open, Collaborative, Community Development



*Ask not what the community is going to do for you,  
ask what you can do for the community*

- P. Butler, March 2019

<http://www.sasview.org>

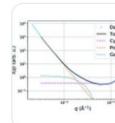
<http://github.com/SasView>



Paavo A. Penttilä  
@PaavoPenttila

Follow

It's there, finally! The main outcome of my postdoc @ILLGrenoble: "Small-angle scattering model for efficient characterization of wood nanostructure and moisture behaviour" And it's all free!



Small-angle scattering  
A small-angle scattering capabilities for studying behaviour of wood are d  
[scripts.iucr.org](http://scripts.iucr.org)

11:33 PM - 26 Mar 2019

1 Retweet 8 Likes



1



8



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1



1

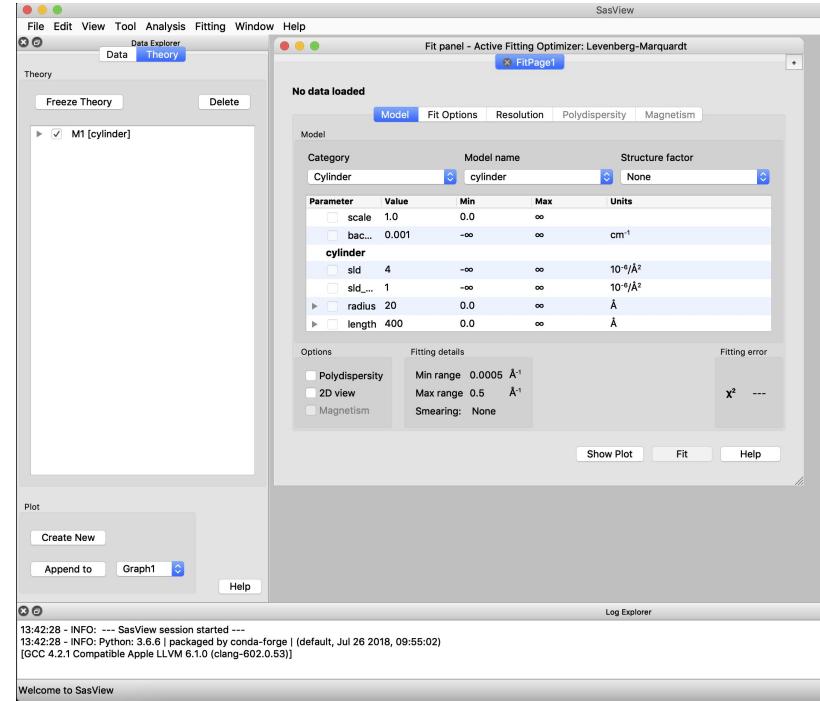
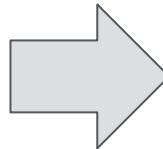
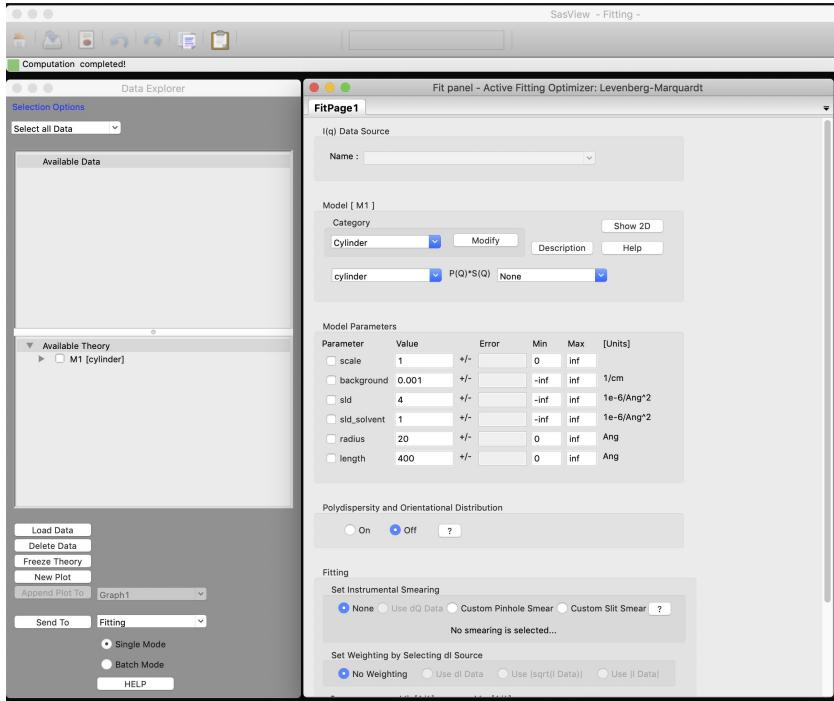


1



1

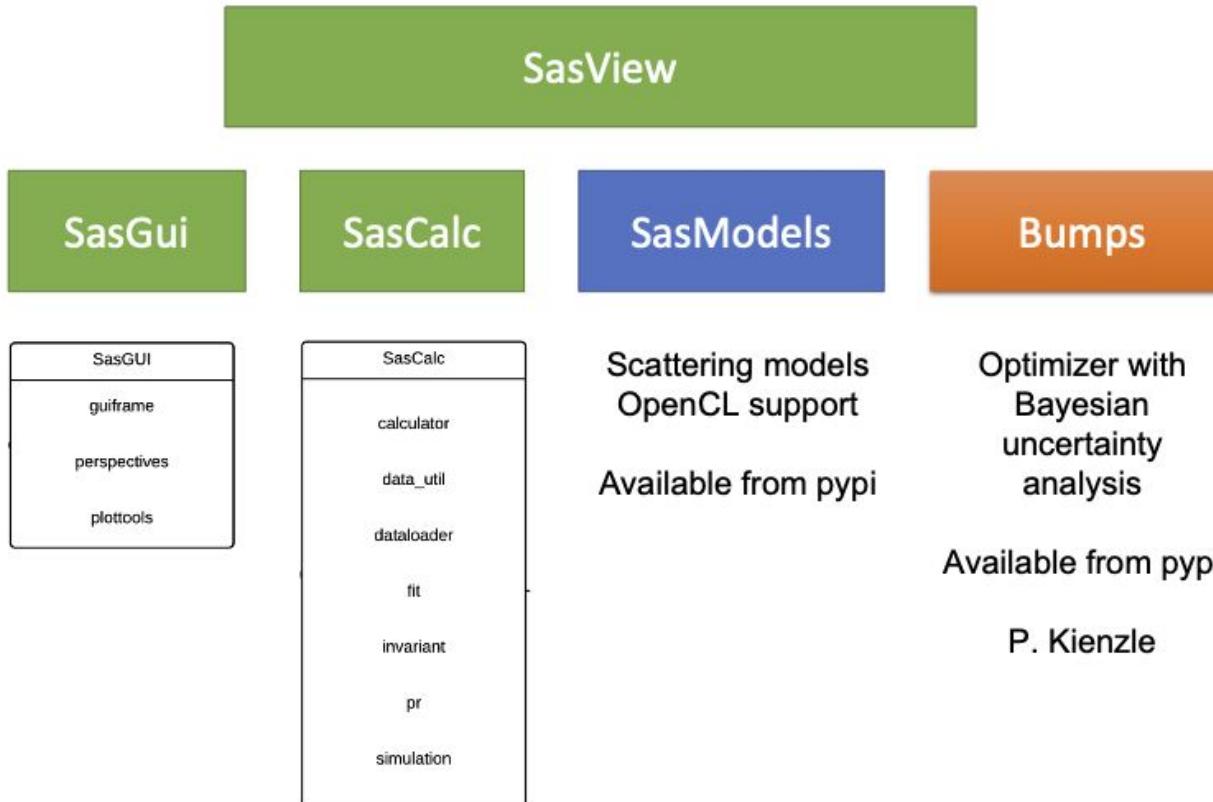
# SINE 2020 Work - GUI



wxPython  
Computation and GUI code mixed  
“Organically developed” - hard for new developers

PyQT  
Computation, GUI code, and models separated  
Structured and documented - easier for new developers

# SINE 2020 Work - Code Separation



# Jupyter Notebooks

## SasCalc example

A simple example demonstrating pair distance distribution function  $P(r)$  inversion. In SasView it is calculated using Moore formula (1980)

```
In [2]: from sas.sascalc.dataloader.loader import Loader
from sas.sascalc.pr.invertor import Invertor
import matplotlib.pyplot as plt
import numpy as np

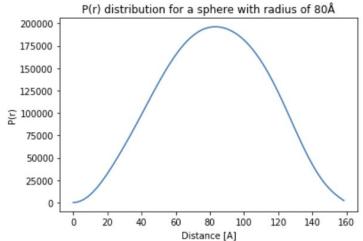
loader = Loader()
test_data = loader.load("sphere_80_err.txt")
x_data = test_data[0].x
y_data = test_data[0].y
z_data = test_data[0].dy

pr = Invertor()
pr.x = x_data
pr.y = y_data
pr.err = z_data

pr.alpha = 2.6e-5
pr.d_max = 160

#nfunc - number of base functions to use.
out, cov = pr.invert(nfunc=13)
pr_value = []
err_value = []
r = np.arange(0.0, pr.d_max, pr.d_max / pr.x.size)
for r_i in r:
    (value, err) = pr.pr_err(out, cov, r_i)
    pr_value.append(value)
    err_value.append(err)

plt.plot(r, pr_value)
plt.xlabel("Distance [Å]")
plt.ylabel("P(r)")
plt.title('P(r) distribution for a sphere with radius of 80Å')
plt.show()
```

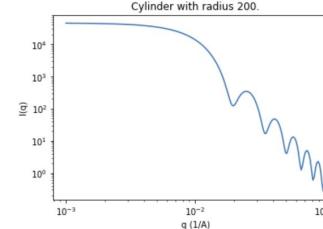


## SasModels example

SasModels is a library of form and structure factor functions. The following example demonstrates how to generate a scattering pattern of a form factor of the cylinder model using sasmodels library. It requires sasmodels to be installed in the path.

```
In [35]: from numpy import logspace
from matplotlib import pyplot as plt
from sasmodels.core import load_model
from sasmodels.direct_model import call_kernel

model = load_model('cylinder')
q = logspace(-3, -1, 200)
kernel = model.make_kernel([q])
Iq = call_kernel(kernel, dict(radius=200))
plt.loglog(q, Iq)
plt.xlabel('q (1/Å)')
plt.ylabel('I(q)')
plt.title('Cylinder with radius 200.')
plt.show()
```



# Jupyter Notebooks

## Fitting model function to data using bumps

The model functions from sasmodels can be used to fit experimental data. This can be done using bumps, which similiar to sasmodels is a separate package and needs to be installed in your path.

```
In [36]: from sasmodels.core import load_model
from sasmodels.bumps_model import Model, Experiment
from sasmodels.data import load_data

from bumps.names import *
from bumps.fitters import fit
from bumps.formatnum import format_uncertainty

import pylab

test_data = load_data('cyl_400_20.txt')
kernel = load_model('cylinder')

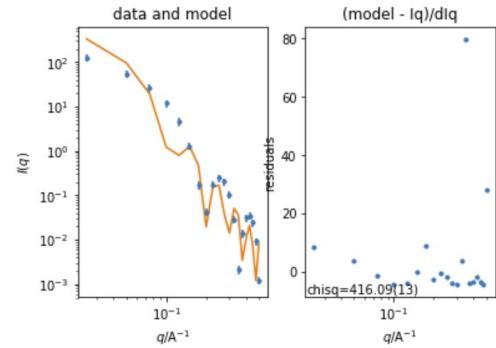
#We set some errors for demonstration
test_data.dy = 0.2*test_data.y

pars = dict(radius=35,
            length=350,
            background=0.0,
            scale=1.0,
            sld=4.0,
            sld_solvent=1.0)
model = Model(kernel, **pars)

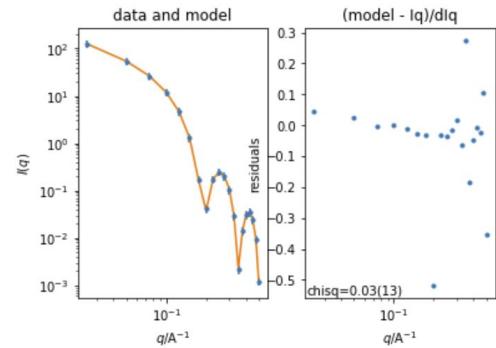
# SET THE FITTING PARAMETERS
model.radius.range(1, 50)
model.length.range(1, 500)

M = Experiment(data=test_data, model=model)
problem = FitProblem(M)
print("Initial chisq", problem.chisq_str())
problem.plot()
pylab.show()

result = fit(problem, method='amoeba')
print("Final chisq", problem.chisq_str())
for k, v, dv in zip(problem.labels(), result.x, result.dx):
    print(k, ":", format_uncertainty(v, dv))
problem.plot()
pylab.show()
```



Final chisq 0.03(13)  
length : 464.9(55)  
radius : 19.977(64)



# The RoadMap

## SasView 5 Year Roadmap

The purpose of building and operating large scattering facilities is to provide unique tools to answer new scientific questions with the final presentation of results (usually in the form of a paper) as the output. The biggest obstacle to that output is often the analysis of the acquired data. Data analysis software has been variously viewed as being in the domain of the scientist using the facility, a service to be provided by scattering facilities, or as the individual responsibility of the scientists running the facility beamlines. The result has been a proliferation of packages and libraries, many written and supported by one key person, often not as their primary responsibility.<sup>1</sup>.

Over the past decade several trends have contributed to exacerbate the analysis bottleneck: 1) As the techniques have matured the user pool has broadened. This combined with an apparent decrease in the overall level of programming taught to scientists, means that fewer users are capable of building their own analysis tools. 2) With the increasing maturity of the field, a large amount of basic modeling is well understood and developed. Even those capable of coding their own should not be wasting their time re-inventing the wheel but focus on new science and perhaps new analysis developments to enable that new science. 3) The quantity of data being produced by instruments and the complexity of the experiments being performed have increased. 4) Finally, as the general software landscape has moved towards increased quality of usability and expectation data.

**Directs work for developers and helps find candidate projects for funding.  
Discussed and updated at each Code Camp.**

### Late 2018 to mid 2019 (from code camp VIII - ESS) - Release 4.2, Release 5.0

The focus in this period will be on development and release of version 5.0 of SasView. In parallel version 4.2 and possibly 4.3 will be released providing a maintained, stable, release for current users of SasView. This managed transition from the 4.x series to the 5.x series will allow for extensive user testing of the 5.0 version prior to release. We expect to continue maintenance of the final 4.x release beyond the release of 5.0, with an eventual end-of-life for 4.x occurring with the 5.2 release.

Full integration of the beta approximation work into 5.0 will be completed, with some limited beta approximation functionality being made available in 4.x.

The first SasView community meeting will be held at the SAS 2018 meeting in October 2018 providing SasView users and contributors with an introduction to the new functionality being made available in 5.0 and training on how to get involved in contributing to the SasView project. Building on this meeting a plan for expanding community interactions will be developed.

Release 4.2 and 5.0 will support separate plotting of the P(O) and S(O) components in a P\*S

### Living document

design

the

# Roadmap Late 2018 to mid 2019

- Move focus of all GUI efforts to the new Qt GUI **Done**. Major bug fixes only to 4.x GUI
  - Parallel development and release tracks (5.x + 4.x) **Working**, but needs streamlining from 5.0 release
  - Complete beta approximation work **In testing**
  - New, more flexible interaction volumes/radii **Underway?**
  - Community meeting at SAS 2018 **Done**
  - Complete SasView paper **Started**
  - Consolidate and extend training material - both written tutorials and hands-on training material. **Ongoing**
  - Update model marketplace **Needs developer**
  - Create plan for developing community interactions. **Started**
  - Fixes to custom model editor to support polydispersity **Done**
  - Incorporation of models from:
    - a. SASFit<sup>^7</sup> **Work done**, but not shipping by default. <https://github.com/SasView/sasfit-models>
    - b. Scatter<sup>^8</sup> (Förster - crystalline materials models primarily) **In discussions with BornAgain team**
  - Project infrastructure cleanup:
    - a. ticket review/cull given 5.0 release **Ongoing**
    - b. possible move to GitHub issues. **Underway**
  - Release
    - a. 5.0 alpha (late 2018) **Done**
    - b. 5.0 beta (early 2019) **Done**
    - c. 5.0 (mid 2019) **On track**
    - d. 4.2 **Done**
- Not in roadmap for this period ...

  - Complete separation of sascalc package / headless usage
- <https://github.com/SasView/documents>

# SasView 4.x series - 4.2.1 current

[www.sasview.org](http://www.sasview.org)

Official Releases available for Windows, Mac, and Debian Linux

## Models

New models

New model package (sasmmodels)

*Separation of models from GUI*

*Simpler addition of models by users*

*Speed! GPU and parallel processing*

*New, consistent approach to orientation distributions for 2D*

## Correlation Function Analysis

*CCP13 corfunc algorithm*

## Documentation

Enhanced, updated documentation for models.

New Tutorials.

## SESANS

Automatic transform of SANS model to P(z)

Plotting and fitting of SESANS data from GUI

Example scripts for fitting SESANS data

Simultaneous fitting of SANS & SESANS



# SasView – the Next Generation – 5.x

**Parallel release of 4.x and 5.x until 5 series is stable.**

**5.0 beta2 release out now for testing!**



## UI Refactoring (“SasView 5.0”)

Move to QT - current and well supported toolkit  
Complete separation of GUI and calculation code  
Provision of CLI & updated Python API

→ Release 5.0

## Sasmodes Enhancements

Return  $F(q)$  from models  
*Beta approximation*  
*Coherent sums*

→ Target release - 5.0



Constraints refactor  
Multi-GPU support  
Inclusion of SasFit models

→ Target release - 5.1

→ Target release - 5.0

Integration of McSAS  
Implementation of key models from Scatter

→ Target release - 5.1

## Documentation

Tutorials – written, interactive & video  
Manual

# Education & Outreach

- Website
- Documentation
- Written Tutorials
- Video Tutorials (YouTube)
- Taught courses
  - Scattering schools
  - University courses
- E-learning
- Bootcamps & Regional Workshops
- Twitter
- Slack
- (Marketplace)

SasView 4.1.2 documentation » SasView User Documentation » Working with SasView »

previous | next | modules

## Table Of Contents

- Loading Data
  - The data explorer
  - Loading data
  - The handy menu
  - Activating data
  - Removing data
  - Creating a new plot
  - Appending plots to a group
  - Freezing the theory
  - Sending data to applications
- Previous topic
- Data Formats
- Next topic
- Plotting Data/Models
- This Page
- Show Source
- Quick search

Go

## Loading Data

### The data explorer

*Data Explorer* is a panel that allows the user more interactions with data. Some functionalities provided by the *Explorer* are also available through the context menu of plot panels or other menus within the application.

Under View in the menu bar, *Data Explorer* can be toggled between Show and Hide by clicking *Show/Hide Data Explorer*.

*NOTE! When Data Explorer is hidden, all data loaded will be sent directly to the current active analysis, if possible. When Data Explorer is shown, data go first to the Data Explorer.*

## Loading data

To load data, do one of the following:

Select File -> Load Data File(s), and navigate to your data;

Select File -> Load Data Folder, which will attempt to load all the data in the specified folder;

Or, in the *Data Explorer* click the button *Load Data*, then select one or more (by holding down the Ctrl key) files to load into SasView.

SasView

FAQ

Here are the answers to some common questions about SasView

What is SasView?

What platforms does SasView run on?

Do I need to install Python/C++ or any compilers before I install SasView?

Can SasView make use of my GPU(s)?

Can I stop SasView trying to use my GPU(s)?

Can I use SasView to analyse SANS/USANS data?

Can I use SasView to analyse SAXS/USAXS data?

Home Notifications Messages

What's happening? View 1 new Tweet

Mantid Project @mantidproject · 1h [mantid] [github.com/mantidproject...](#) Owen Arnold - 7 commits

mantidproject/mantid Main repository for Mantid code. Contribute to mantid development by creating an account on GitHub. [github.com](#)

SSI - software.ac.uk @SoftwareSaved · 2h Fixing typo: Thanks to Carole Gobie, Christopher Arnott and John Wise for speaking #BBSRCIscience today!

SSI - software.ac.uk @SoftwareSaved · 2h Thanks to Carole Gobie, Christopher Arnott and John Wise for speaking #BBSRCIscience today!

Mantid Project @mantidproject · 2h [mantid] [github.com/mantidproject...](#) Anton Picard-Selg - 1 commits

Translate from Estonian

mantidproject/mantid Main repository for Mantid code. Contribute to mantid development by creating an account on GitHub. [github.com](#)

# Education & Outreach

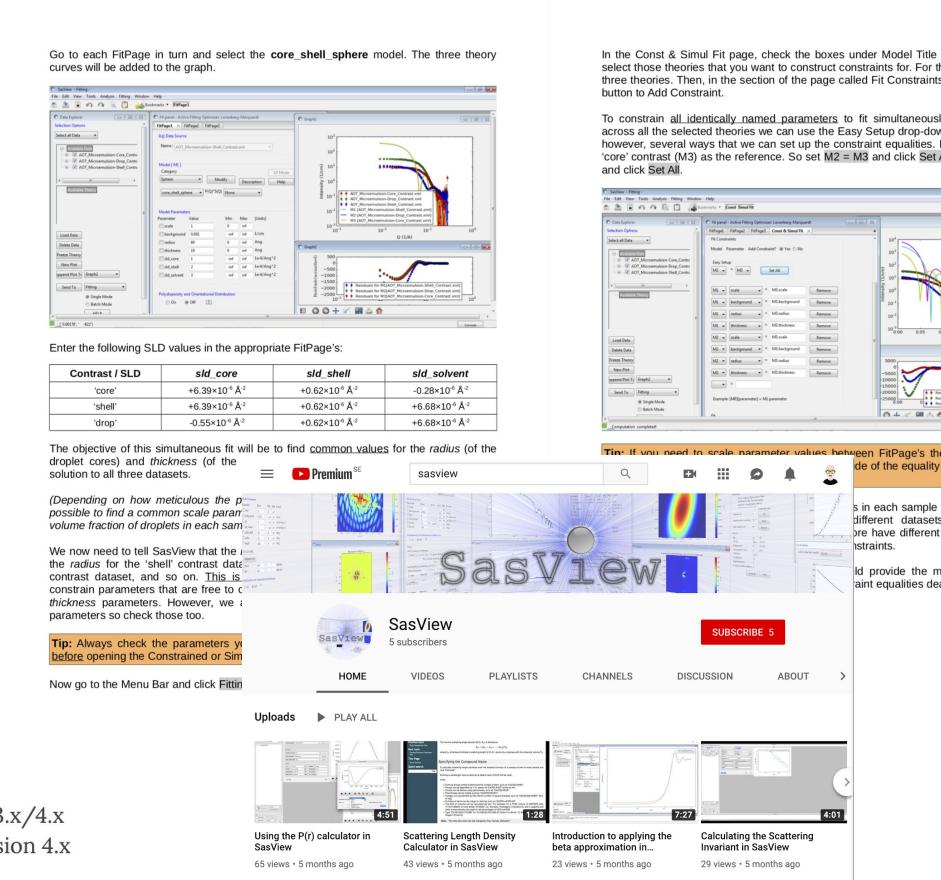
- Website
- Documentation
- Written Tutorials
- Video Tutorials (YouTube)
- Taught courses
  - Scattering schools
  - University courses
- E-learning
- Bootcamps & Regional Workshops
- Twitter
- Slack
- (Marketplace)

**SasView**

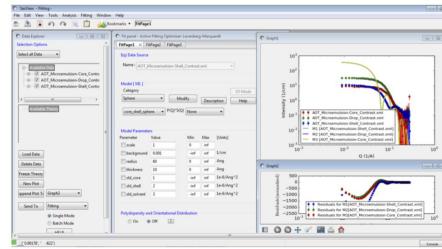
ABOUT ▾ LINE

Tutorials:

- Old SasView tutorial (PDF) - still useful
- Getting started with SasView (PDF)
- Basic 1D Fitting in SasView (PDF) - for versions 3.x/4.x
- Simultaneous 1D Fitting in SasView (PDF) - for versions 3.x/4.x
- Correlation Function Analysis in SasView (PDF) - for version 4.x



Go to each FitPage in turn and select the `core_shell_sphere` model. The three theory curves will be added to the graph.



Enter the following SLD values in the appropriate FitPage's:

Contrast / SLD	<code>sld_core</code>	<code>sld_shell</code>	<code>sld_solvent</code>
'core'	$+6.39 \times 10^{-6} \text{ \AA}^2$	$+0.62 \times 10^{-6} \text{ \AA}^2$	$-0.28 \times 10^{-6} \text{ \AA}^2$
'shell'	$+6.39 \times 10^{-6} \text{ \AA}^2$	$+0.62 \times 10^{-6} \text{ \AA}^2$	$+6.68 \times 10^{-6} \text{ \AA}^2$
'drop'	$-0.55 \times 10^{-6} \text{ \AA}^2$	$+0.62 \times 10^{-6} \text{ \AA}^2$	$+6.68 \times 10^{-6} \text{ \AA}^2$

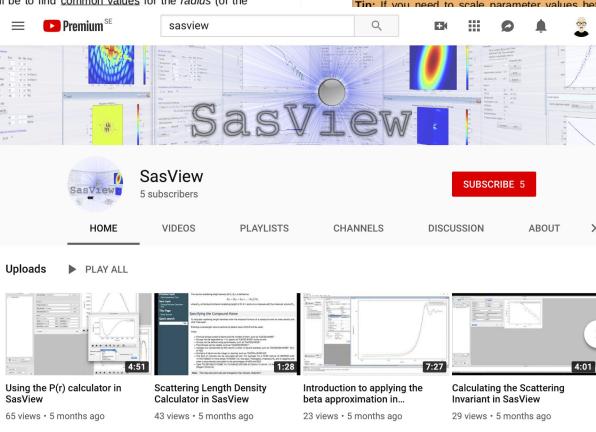
The objective of this simultaneous fit will be to find common values for the radius (of the droplet cores) and thickness (of the shell) to all three datasets.

(Depending on how meticulous the parameters are, it may be possible to find a common scalar parameter that applies to all three datasets, such as the volume fraction of droplets in each sample.)

We now need to tell SasView that we want to fit the radius for the 'shell' contrast dataset, contrast dataset, and so on. This is done by constraining parameters that are free to vary to have the same value as thickness parameters. However, we have some parameters so check those too.

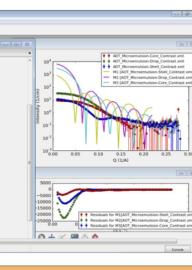
**Tip:** Always check the parameters you are fitting before opening the Constrained or Simulate pages.

Now go to the Menu Bar and click **Fitting**.



In the Const & Simul Fit page, check the boxes under Model Title (or just **Select all**) to select those theories that you want to construct constraints for. For this example, check all three theories. Then, in the section of the page called Fit Constraints, check the **Yes** radio button to Add Constraint.

To constrain all identically named parameters to fit simultaneously to the same value across all the selected theories we can use the **Easy Setup** drop-down buttons. There are, however, several ways that we can set up the constraint equalities. Here we shall use the 'core' contrast (M3) as the reference. So set M2 = M3 and click **Set All**. Then set M1 = M3 and click **Set All**.



Tip: If you need to scale parameter values between FitPage's then use the free-form editor. The equality can be of the form:

$\frac{\text{parameter}_1}{\text{parameter}_2} = \text{constant}$

If each sample was different, remove the constant. If different datasets represent samples that have different incoherent scattering constraints, provide the most sensitivity to the constraints dealing with thickness to the datasets.

Id provide the most sensitivity to the constraints dealing with thickness to the datasets.

# Education & Outreach

- Website
- Documentation
- Written Tutorials
- Video Tutorials (YouTube)
- Taught courses
  - Scattering schools
  - University courses
- E-learning
- Bootcamps & Regional Workshops
- Twitter
- Slack
- (Marketplace)

The screenshot shows the e-neutrons website interface. At the top, there is a navigation bar with links to FRONTPAGE, ABOUT E-NEUTRONS, FOR TEACHERS, and SUPPORT. There is also a login form for Username and Password, and a link to request an account.

**Exercise taster:** This module features a problem titled "Problem: Fourier transform". It includes a text description, a diagram of a 1D scattering system, and a question asking for the normalized scattering intensity. A "READ MORE" button is present.

**Quiz taster:** This module contains a section titled "NEUTRON PROPERTIES" with a question about what neutrons are good for and why. It also has a "READ MORE" button.

**Simulation taster:** This module displays a simulation setup for small-angle scattering. It shows a neutron source emitting a beam onto a sample containing spheres. Below the diagram, there is a table of parameters for the simulation and a section titled "SMALL ANGLE SCATTERING" with a similar question and "READ MORE" button.



All the work of ISIS Sandwich Student Michael Oakley

# Education & Outreach

- Website
- Documentation
- Written Tutorials
- Video Tutorials (YouTube)
- Taught courses
  - Scattering schools
  - University courses
- E-learning
- Bootcamps & Regional Workshops
- Twitter
- Slack
- (Marketplace)

KEMM 37 / EXTN85 / NAKE017 Scattering Methods Computer Lab Guide - 2019

## Lab 1A. Familiarisation with SasView, Geometrical Models, and Structure Factors

This exercise will introduce you to analysing SANS data using geometrical models in SasView. In the first lab session you will look at how different shapes produce different scattering patterns, and how the model parameters affect the scattering pattern. In the second lab session you will then load some real SANS data and attempt to fit models to the data in SasView.

This first exercise is divided into 3 sections:

1. Familiarisation with SasView
2. Exploring geometrical models of form factors
3. Exploring structure factors

## Meeting agenda



- 2:15pm Welcome and intro (goals and outline), Andrew Jackson, 20min
  - What is SasView
  - What is SasView structure: sasview, sasmodes, bumps
- 2:35pm Demo of existing functionality, Andrew Jackson, Paul Butler and Piotr Rozyczko, 1h
  - Going through menu items
  - Loading different data types (1D/2D) data
  - Fitting 1D and 2D models
  - Simultaneous, constrained and batch fitting
  - Calculators
  - Pt inversion, Invariant perspective
  - Correlation functions
- 3:35pm Break 25min
- 4:00pm How to write and distribute user models, Tim Snow, 30min
  - Writing models using plugin editor
  - Category manager
  - Python and C model
  - Distributing models on SasView marketplace
- 4:30pm SasView CLI, Wojtek Potrzebowksi, 15min
  - SasCalc example
  - Calculating form factors from sasmodes
  - 1D fitting using sasmodes and bumps
  - 2D fitting
  - Batch fitting
- 4:45pm Documentation, Tutorials and Bug reporting, Paul Butler, 10min
- 4:55pm How to become a SasView Developer, Paul Butler, 5min
- 5:00pm Community discussion and feedback

# Education & Outreach

- Website
- Documentation
- Written Tutorials
- Video Tutorials (YouTube)
- Taught courses
  - Scattering schools
  - University courses
- E-learning
- Bootcamps & Regional Workshops
- Twitter
- Slack
- (Marketplace)

Uses mySQL & Postgress

8 models contributed by  
7 authors in 2 years

*All the work of ISIS Summer Student Lewis O'Driscoll*



Name	Description	Category	Upload Date	Author
correlated_spheres	Definition ----- The 1D scattering intensity of two correlated spherical particles can be written as: $S(q) = F_1^2 + F_2^2 + 2 * F_1 * F_2 * \sin(qD) / qD$ , where $F_1$ and $F_2$ are the scattering ...	Sphere	30 Mar 2019	Tianfu
WoodSAS	This model is tailored for fitting the equatorial intensity profile from wood samples (Penttilä et al., 2019). The model consists of three independent contributions: 1) Scattering in the plane per...	Cylinder	15 Mar 2019	penttila
Nanodisc	This is a simple re-parameterisation of the core-shell bicelle model such that it can be more easily applied to the fitting of a phospholipid nanodisc.	Cylinder	02 Dec 2018	arm61

# Come and Join the Fun!



## Things people are saying about SansView/SasView

- 'SansView is a very helpful tool, very complete and easy to use' - Niki
- 'I want to thank you for this amazing software. It's UI and options make the interpretation of spectra easier and faster' - Philippe
- 'I really like the SasView software' - Martin
- 'I have been using SasView as my software of choice for fitting SANS data, and I have been very happy with the software' - Greg
- 'I have found SasView very easy to use and the batch fit function is a wonderful time saving tool. I can finally stop making painful excel macros!' - Andrew
- 'I am a new user of SasView and I think it is a very useful and practical tool' - Arnaud
- 'Within 30 seconds...I am completely converted to SasView!' - Mike
- 'Thank you for creating and maintaining SasView. It is an incredibly helpful tool, and I use it regularly' - Pasha
- 'All the best and thank you again to carry on such a good job on SasView' - Niki
- 'Ooooh NICE PROGRAMME!! Hours of fun!' - Stuart
- 'I love such amazing software so much. It help our researches a lot.' - Po-Wei

