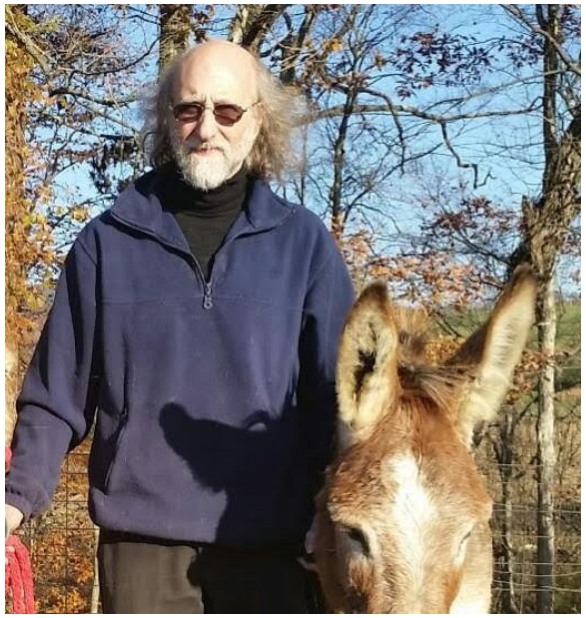
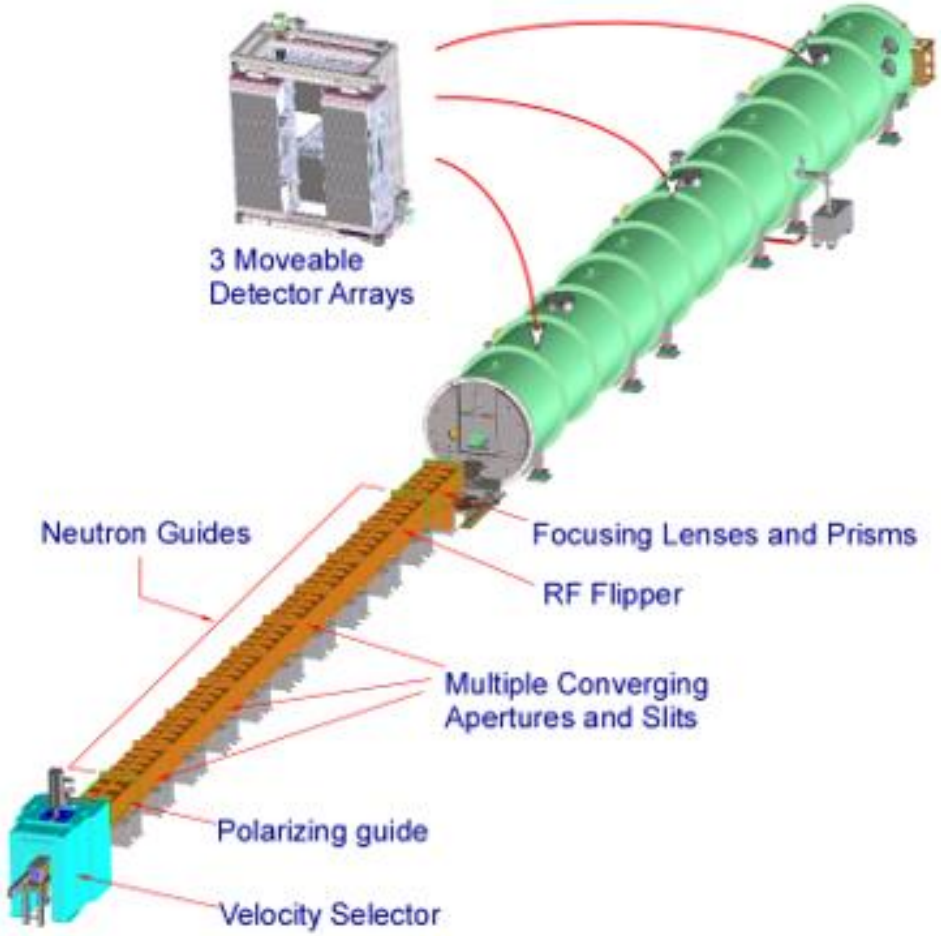

Overview: Current structure, functionality, and Community Development Model
SasView
ISIS
NIST
diamond
OAK RIDGE National Laboratory
ansto
NEUTRONS FOR SCIENCE
ess
TU Delft

# Notes from the Speaker (my 5 min)



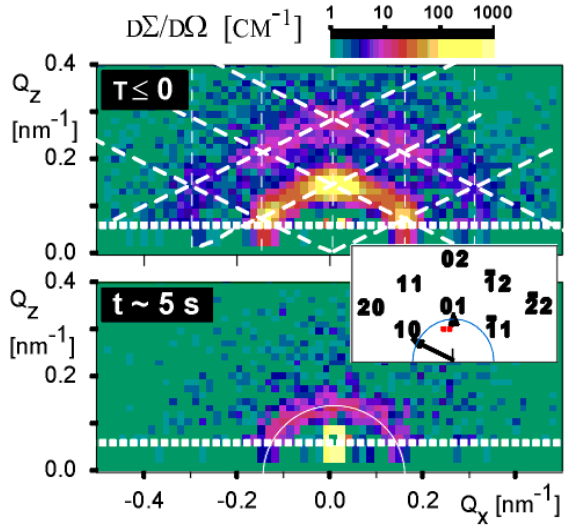
- Scattering since last millennium
- Leader of the NIST small angle group since 2004




# Notes from the Speaker (my 5 min)

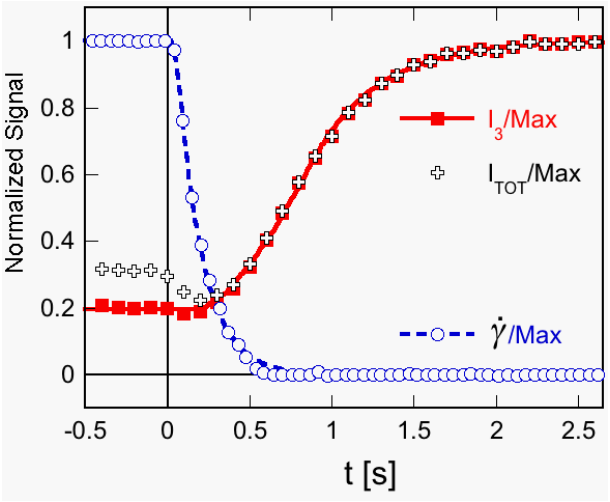
- Colloid/self-assembly scientist

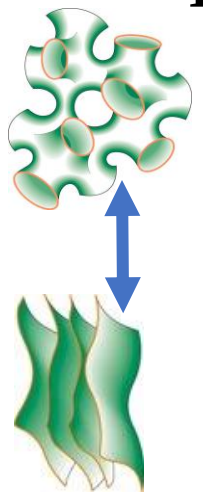
**Ordering of wormlike micelles near a surface and under flow**



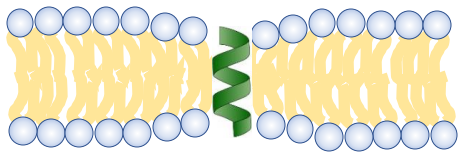

**Flow induced sponge to lamellar transition and relaxation**

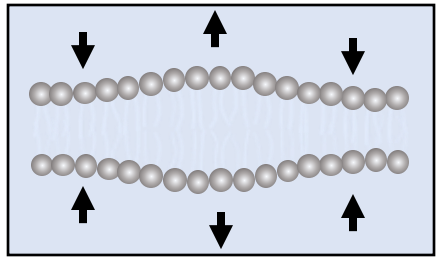
**Energetics of fusion**

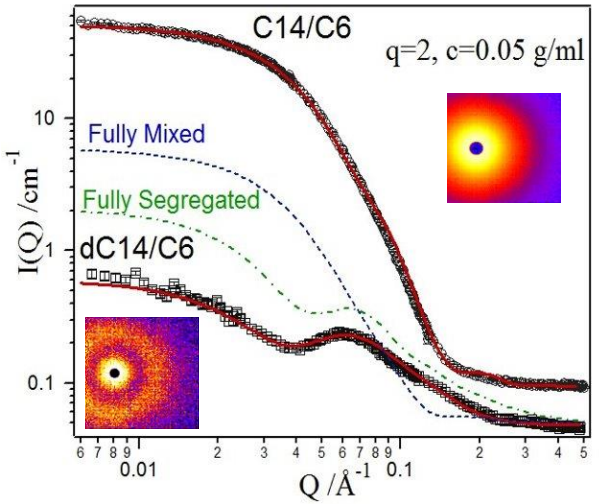


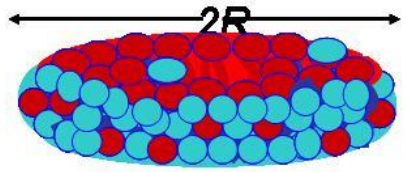



**Lipid membrane structure and dynamics**








**Segregation of lipids in mixed lipid system**

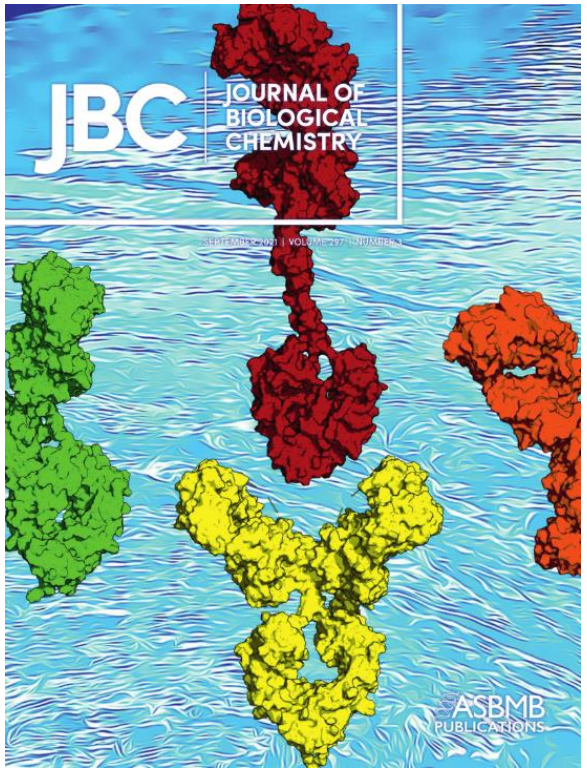
# Notes from the Speaker (my 5 min)

- Long standing interest in building SAS and scattering community and collaboration
  - SasView
  - canSAS
  - CCP-SAS

**canSAS V**

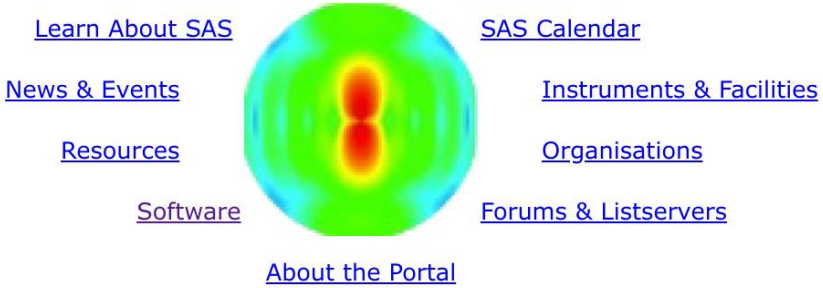
**Workshop Report**

The Fifth canSAS meeting was held on October 29th to 31st on the NIST campus in Gaithersburg, Maryland, hosted by the NCNR SANS group. The aim of the international organizing committee was to keep the group to a reasonable size of motivated people to have frank and open discussion of issues and work on solutions that would be outcomes of the meeting. The format included oral presentations, posters, and software demos as well as plenty of discussion time from moderated discussions, to open discussions, to various informal discussion opportunities. The workshop was successful at stimulating significant levels of interactions and lively discussions.




**SAS 2022 Campinas, Brazil: September 11-16, 2022**

*Mark your calendars*




canSAS V (2007)

- Reduced formats (NXcanSAS)
- Inter-facility standards
- SAS portal (smallangles.org)
- IGOR collaborations
- Facility rep quarterly meetings

# **SasView:** *A SAS data analysis platform*

Data Analysis eh?
…. So what exactly does that mean?

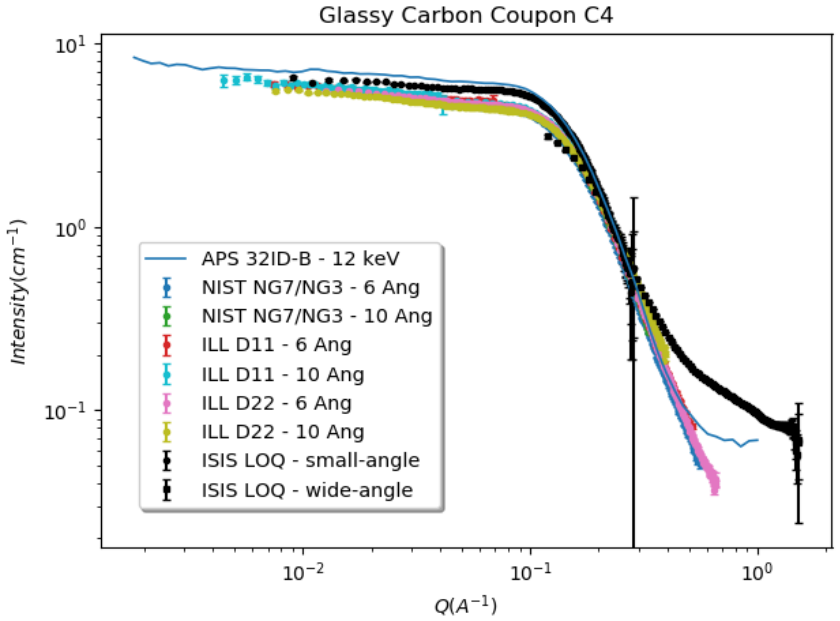
Only works on Reduced data

(All the instrumental artifacts are removed and only the science is left)

….
Sorta

# Experimental Data Flow

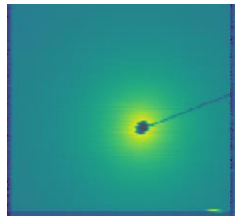
Reduced data should be reproducible anywhere!




http://www.cansas.org

Instrument-*dependent* data

Instrument-***in****dependent* data

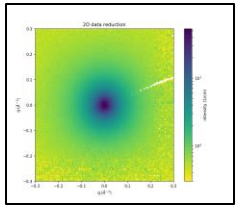


**Data Acquisition System**

'raw' data file (NXS)

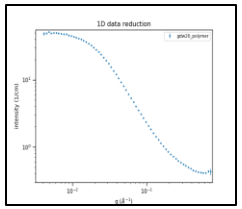
**Data Reduction Programs (e.g. *Mantid*)**

reduced data file (1D: HDF / XML / TXT; 2D: HDF / TXT)



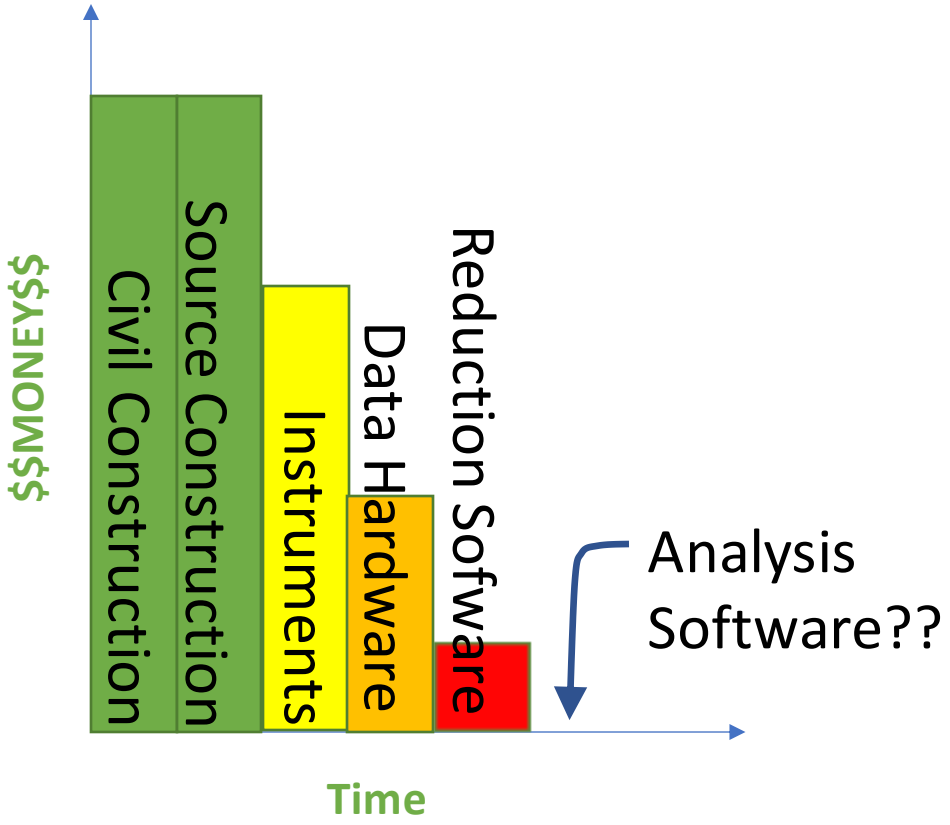
**Data Analysis Programs**

**Paper / Thesis**



# Analysis Software – Who's job is it anyway

# The resource Problem Part I




## Analysis Software - Who's Job is it Anyway?

Scattering is an analysis tool and part of providing the tool should be the necessary software

→ the FACILITY'S JOB

Analysis is where the science is

→ the USER'S JOB

Or maybe …..

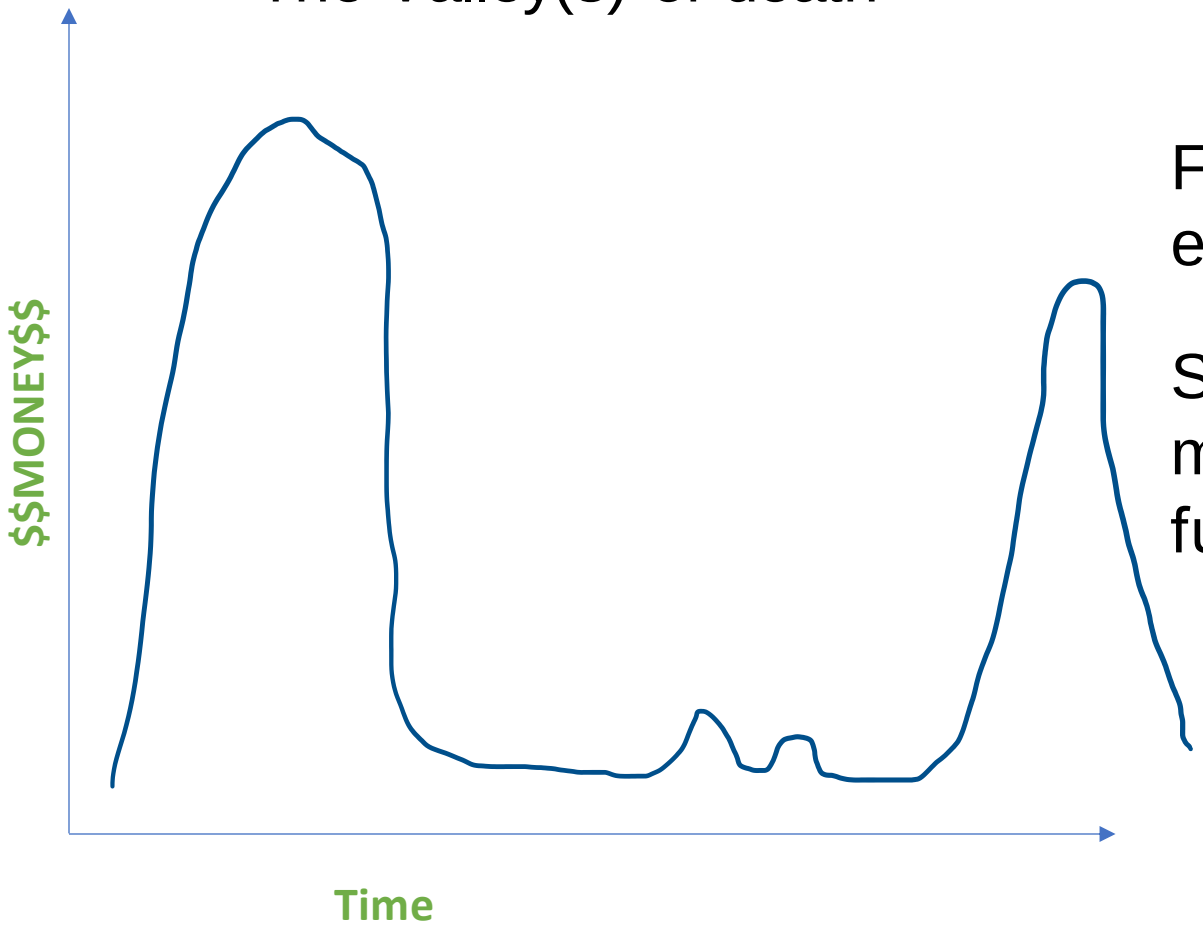
→ The Domain Science funding agencies?

# The resource Problem Part II
# The feast and famine roller coaster

## Analysis Software - Who's Job is it Anyway?

*- The domain science funding agencies*

The Valley(s) of death

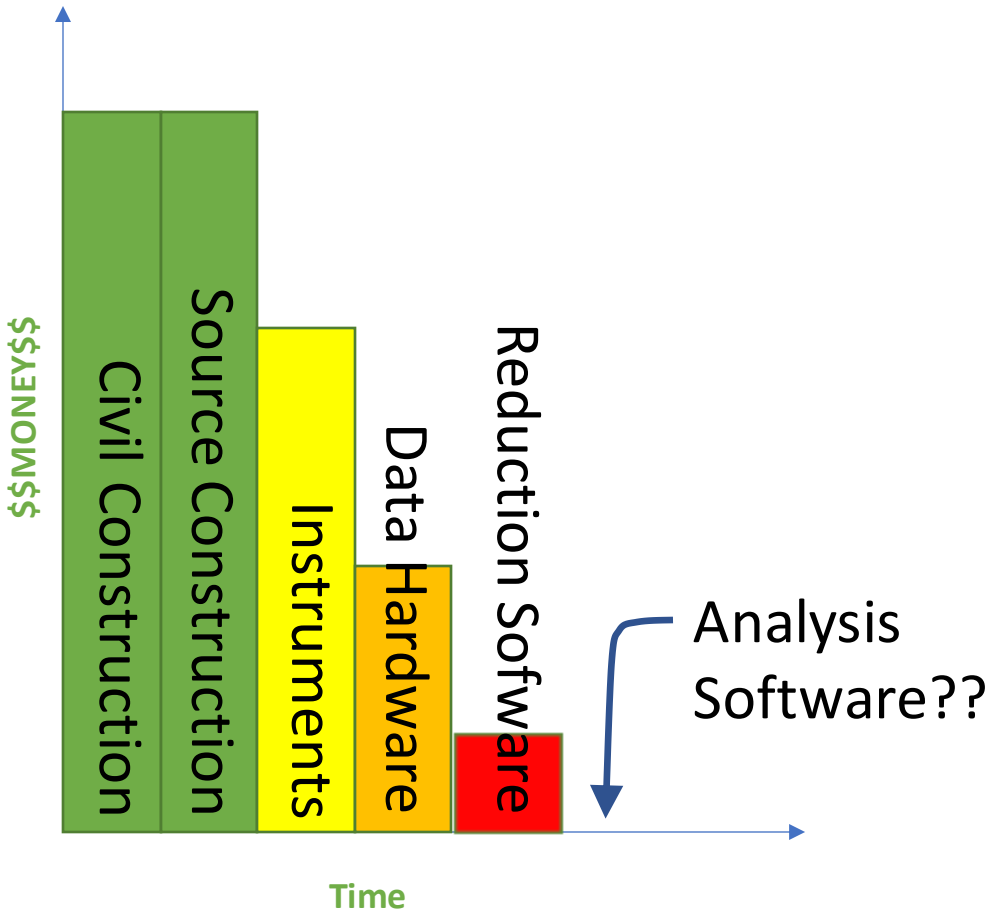



Facility directors discretion, NIH, NSF, DOE, etc.

Special funding (grants) do no fund long term maintenance and ongoing development. They fund "big new (transformative) ideas"

# The resource Problem Part III

## The unbounded problem




Fundamentally all these elements are relatively well defined problems…

EXCEPT ...

Analysis Software is really unbounded. The needs are nearly infinite and constantly evolving.

HOWEVER --- Analysis as defined here is also uniquely universal and ripe for collaborative pooling of finite resources

…. But beware the monoculture?

# The resource Problem Reframed – The Need

- A way to focus limited resources on top priorities (most useful to the science being done) in a world of infinite possibilities
- A way to harness funding for bold new ideas without losing the investment in the valleys of death
- A way to provide sustained maintenance and development in an uncertain funding environment (thriving through the famines)

**FACTS OF LIFE:**

- **Resources are finite**
- **Needs are infinite**


# The SasView Aproach

An
"open, collaborative, community development"
platform for
Small Angle Scattering Data Analysis

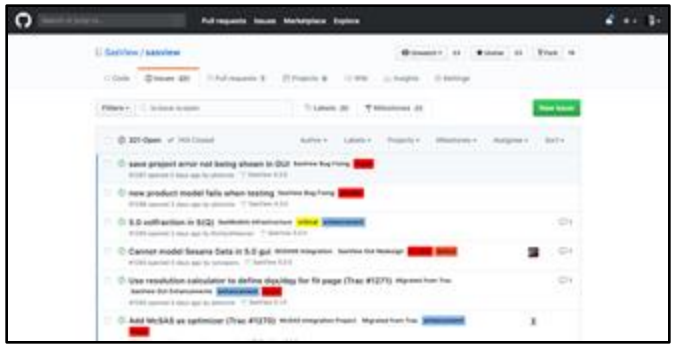

# **Open**, Collaborative, Community **Development**

Code is open source and publicly hosted at Github

Released under BSD 3-clause license

(Zenodo) DOI for **each release**

Code Hosting, **Issue Tracking,** Developer Wiki & CI on Public Github repos



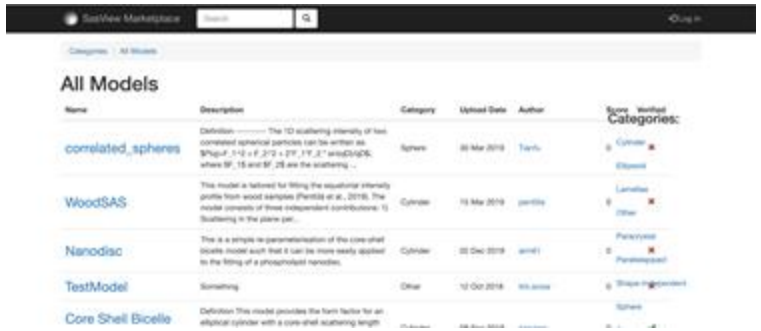
https://github.com/SasView

- Website
- Documentation: in-program & online
- Written Tutorials
- Video Tutorials (YouTube)
- scattering schools/workshop
- university courses
- Bootcamps & regional workshops
- Twitter

- help@sasview.org
- users@sasview.org

Rolling **5 Year Roadmap**

**Model Marketplace** for Users to share their models




http://marketplace.sasview.org

python™

**https://www.sasview.org**

matplotlib

# Open, **Collaborative**, Community **Development**

- Twice monthly zoom calls
- Regular ‘camps’ & ‘hackathons’
- Developer's mailing list
- SasView slack
- Expertise sharing and helping
- Small leadership team to facilitate

**Collaboration also builds Community**

http://www.sasview.org

http://github.com/SasView

# 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

*No MOU ... all are invited and welcome*

*Two Basic "Rules"*

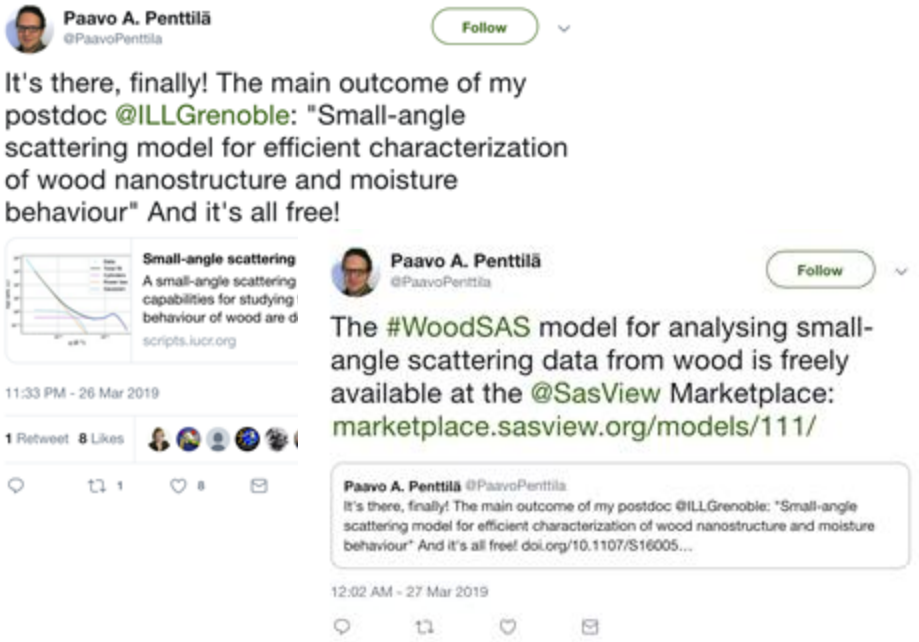
*He who pays the piper ...*

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

*And ...*

*You cannot break existing experiences ...*

- New dependencies vs long term maintenance (sustainability)
  - Code quality vs long term sustainability
  - Changing/degrading the current user experience for the existing user base

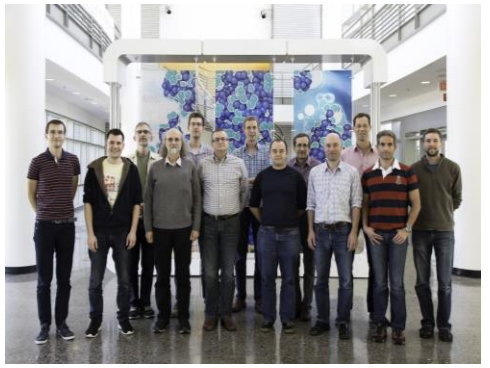


# History and Status of SasView

- 2006; originates in NSF *DANSE* project
- 2013; transitions into a community project
- 2016; Sine2020 project funded
- 2022; Essentially a “volunteer army”

---

- ~40 contributors from 9 organizations so far (~10-15 active at any one time)
- 1 to 2 releases/year (5.0.5 JUST RELEASED)
- Documentation/tutorial projects ongoing
- Usage? Seems to be "everywhere?"
- Publications? > 100/year



# SasView Model: Some Current Problems

## The price of success:

- Many people view the project as a well-funded group of professionals → barrier to contribution – e.g: Old website and no graphics = not serious; Nice graphic slick website = professional → way beyond my skills – how to navigate?
- Large project with many moving pieces is a barrier to new volunteer coders
- Heavily geared towards colloid science
- The tragedy of the commons: somebody else will take care of the problem.
- Hard to get non-coders to believe they can contribute equally – it is unfortunately in the name "code camp" … words matter → SasView camp?
- Building community, especially during a pandemic is HARD WORK
- Lack of funds for small things … but beware too much money?
- Hard to keep up with increasing security issues with these resources so far.

# The resource Problem Reframed – The Solution

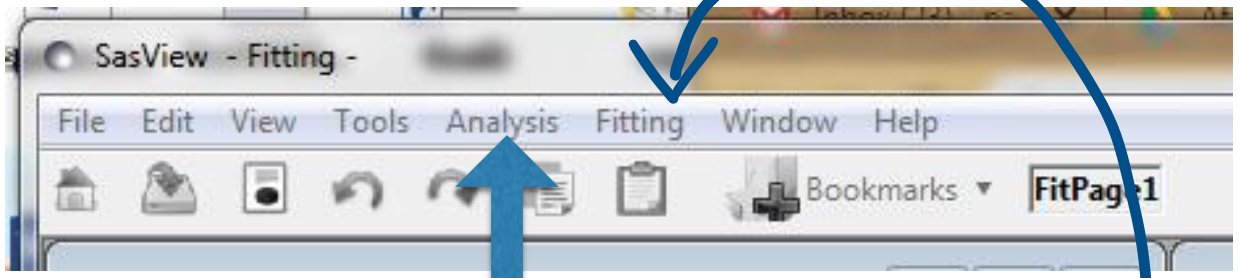
## The SasView approach

- Facilities provide foundational support through participation of data and instrument scientists
- Grants and other projects provide “bold new functionality”
- Community, writ large, helps provide support and functionality
- New ideas tested and developed as before by individuals or larger groups (the community)
- Once validated and deemed ready for the larger community these groups provide resources (their labor) to integrate while active developer community helps with training on where things go and on parts of interest to them (collaborative)

So …

What Can SasView Do Currently?

# Perspectives on the data




## Tools

- Data Operation
- SLD calculator
- Density/Volume calculator
- Slit Size Calculator
- Kiessig Thickness Calculator
- Q Resolution Estimator
- Generic Scattering calculator
- Orientation Viewer
- Python Shell/Editor
- Image Viewer
- File Converter

## Analysis

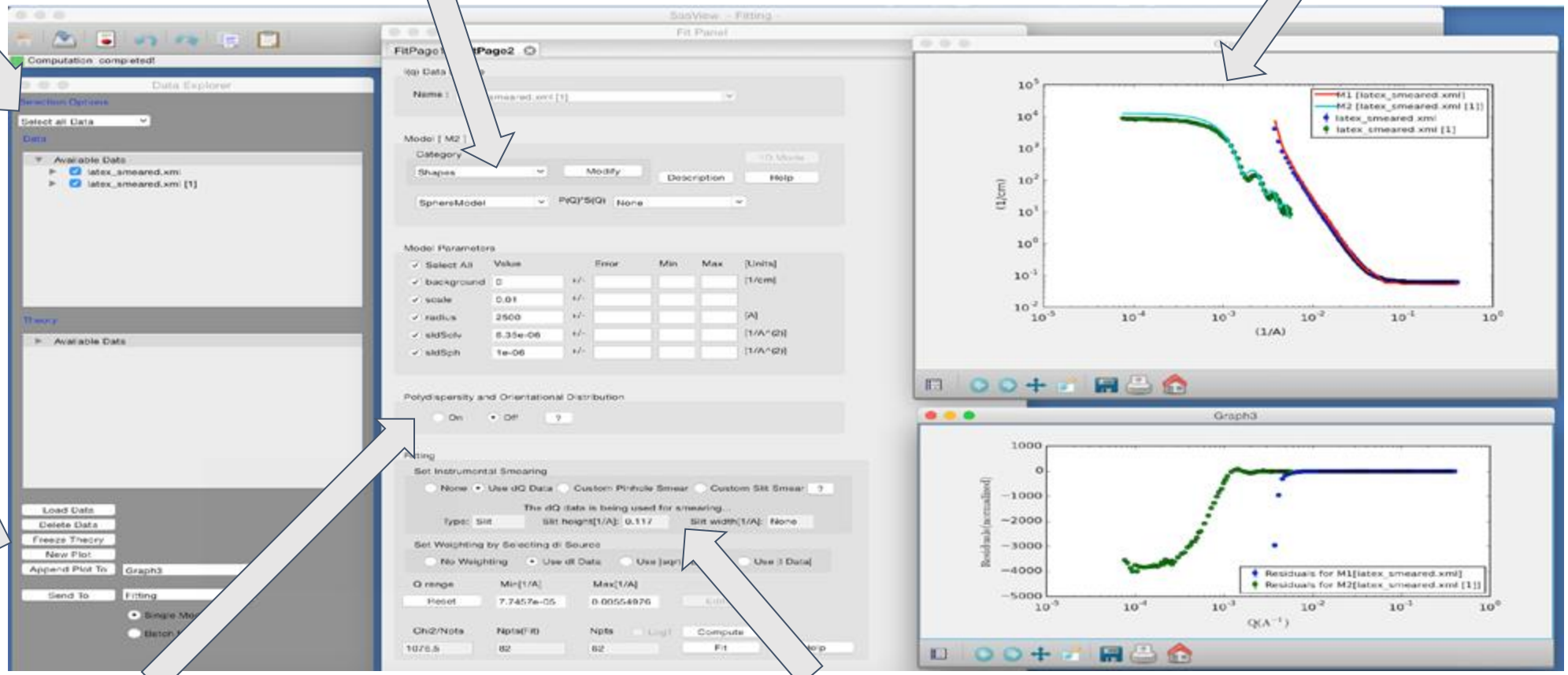
- Fitting
- Invariant
- Pr Inversion
- Correlation Function

# Fitting Perspective: 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)

Simultaneous fitting

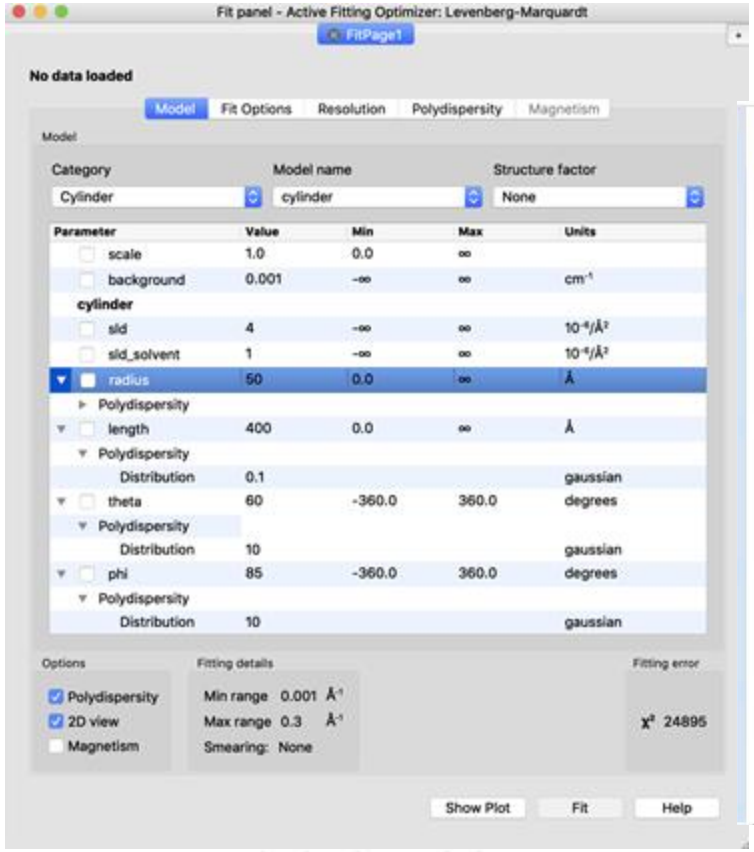


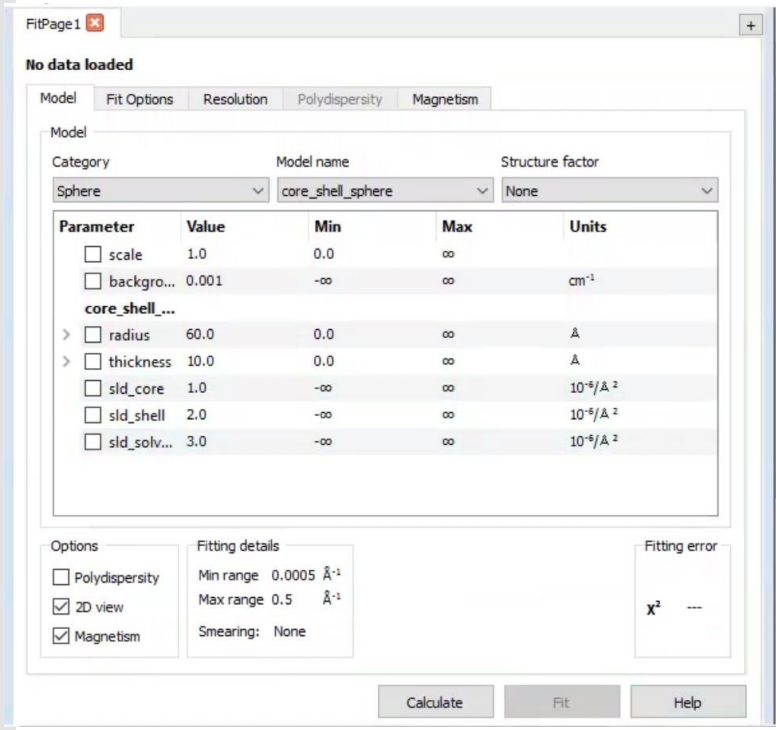

Analysis Tool Choice & Plotting

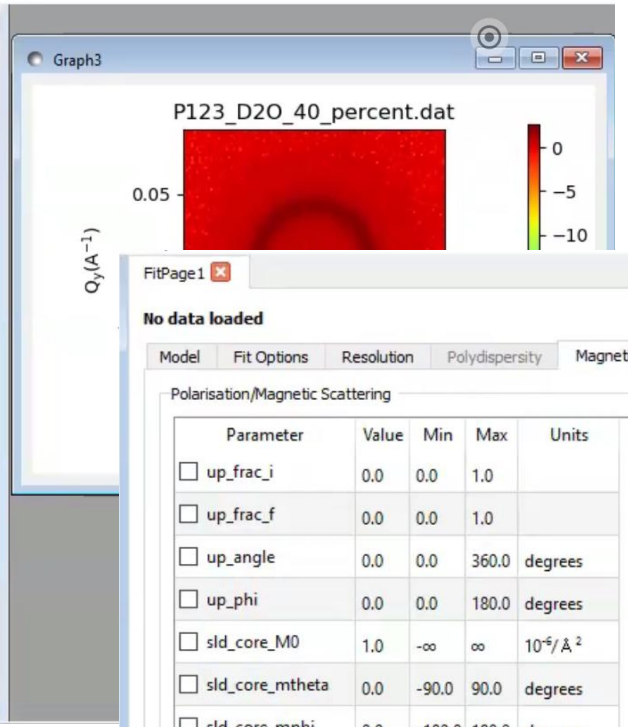
Generic parameter polydispersity
Choice of distribution and distribution parameters

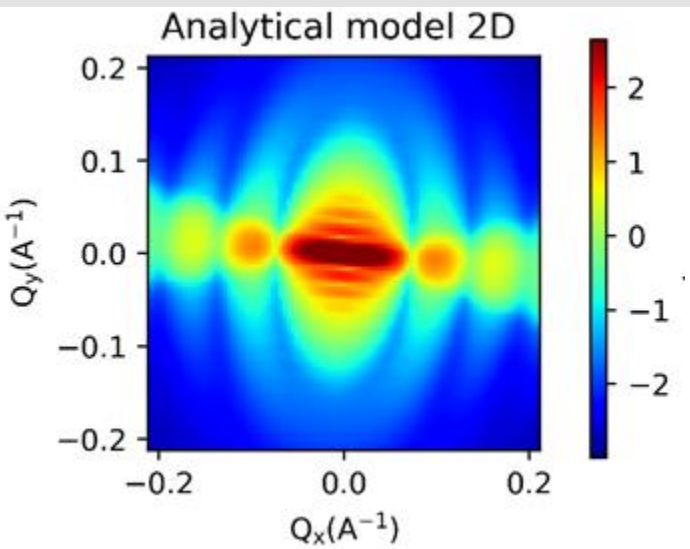
Resolution smearing (pinhole and slit)
Automatically from data or provide parameters

# Fitting Perspective: 2D Analysis




Orientational polydispersity = “jitter”

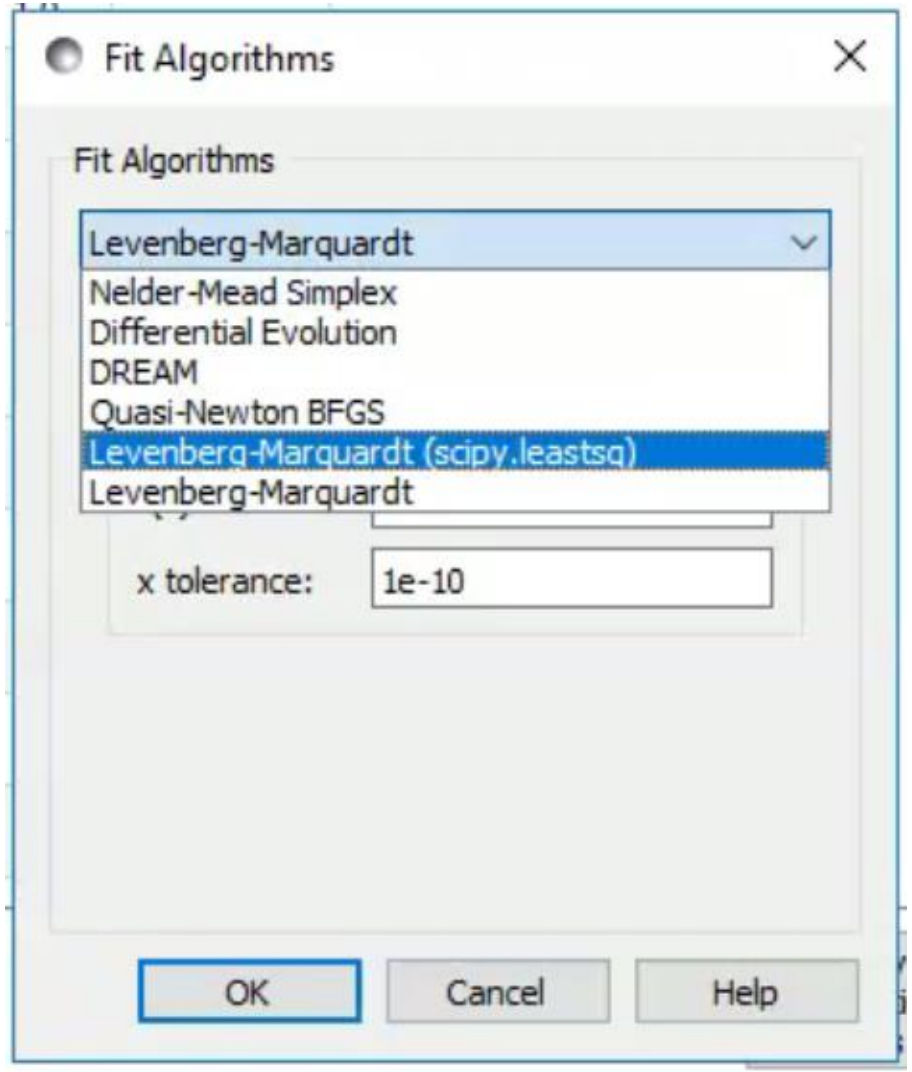
Decouples the frame for the object’s orientation with respect to the beam and the “jitter” around the axis of the object.

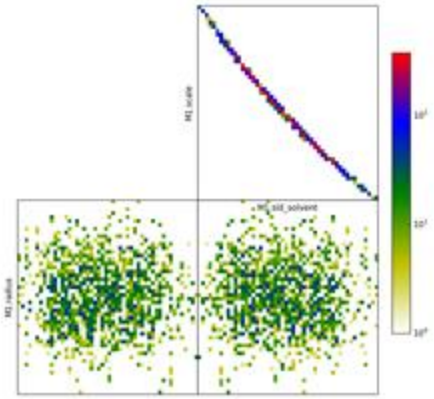
Turning on GPU Option highly recommended for fitting

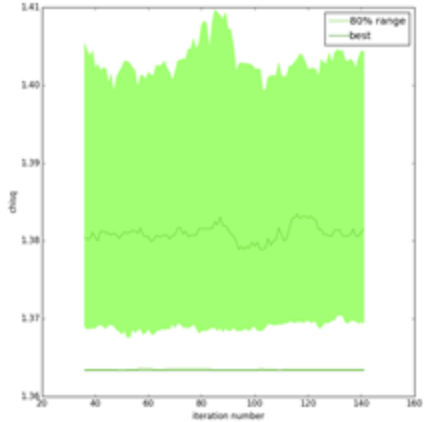
# Choice of Fitting Algorithms

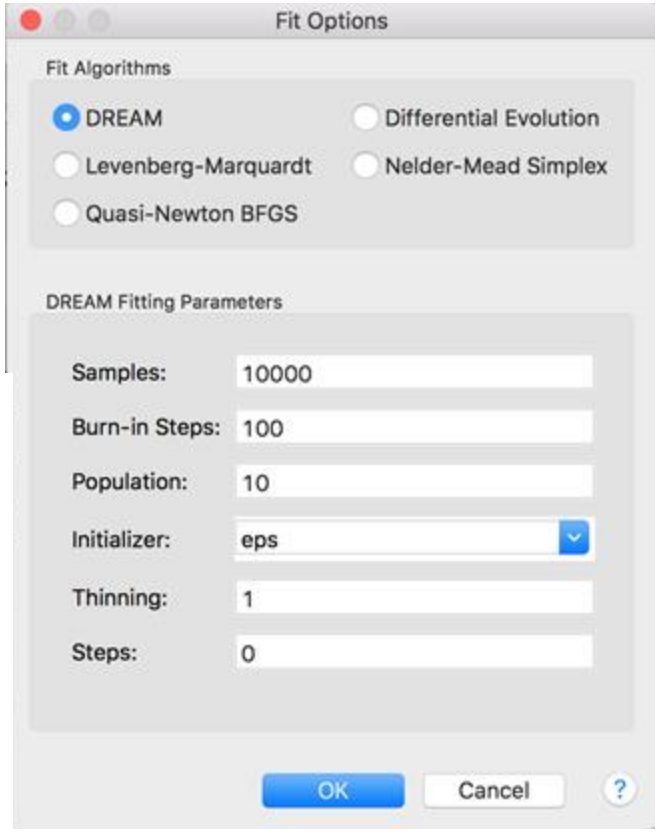
Uses bumps package from P. Kienzle

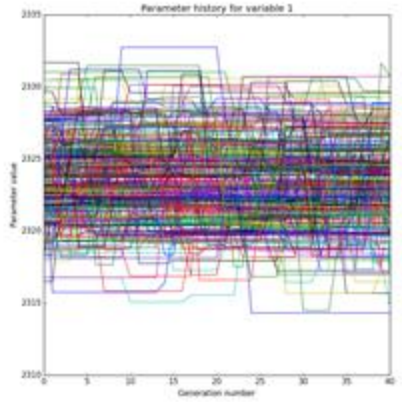
(also has DANSE origins)

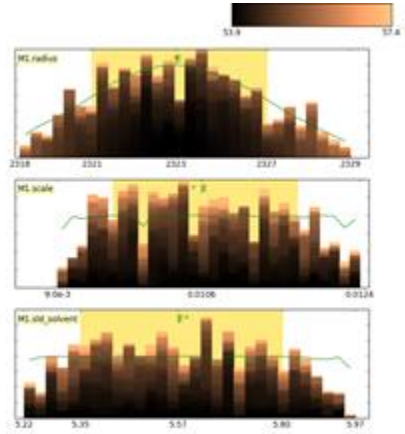


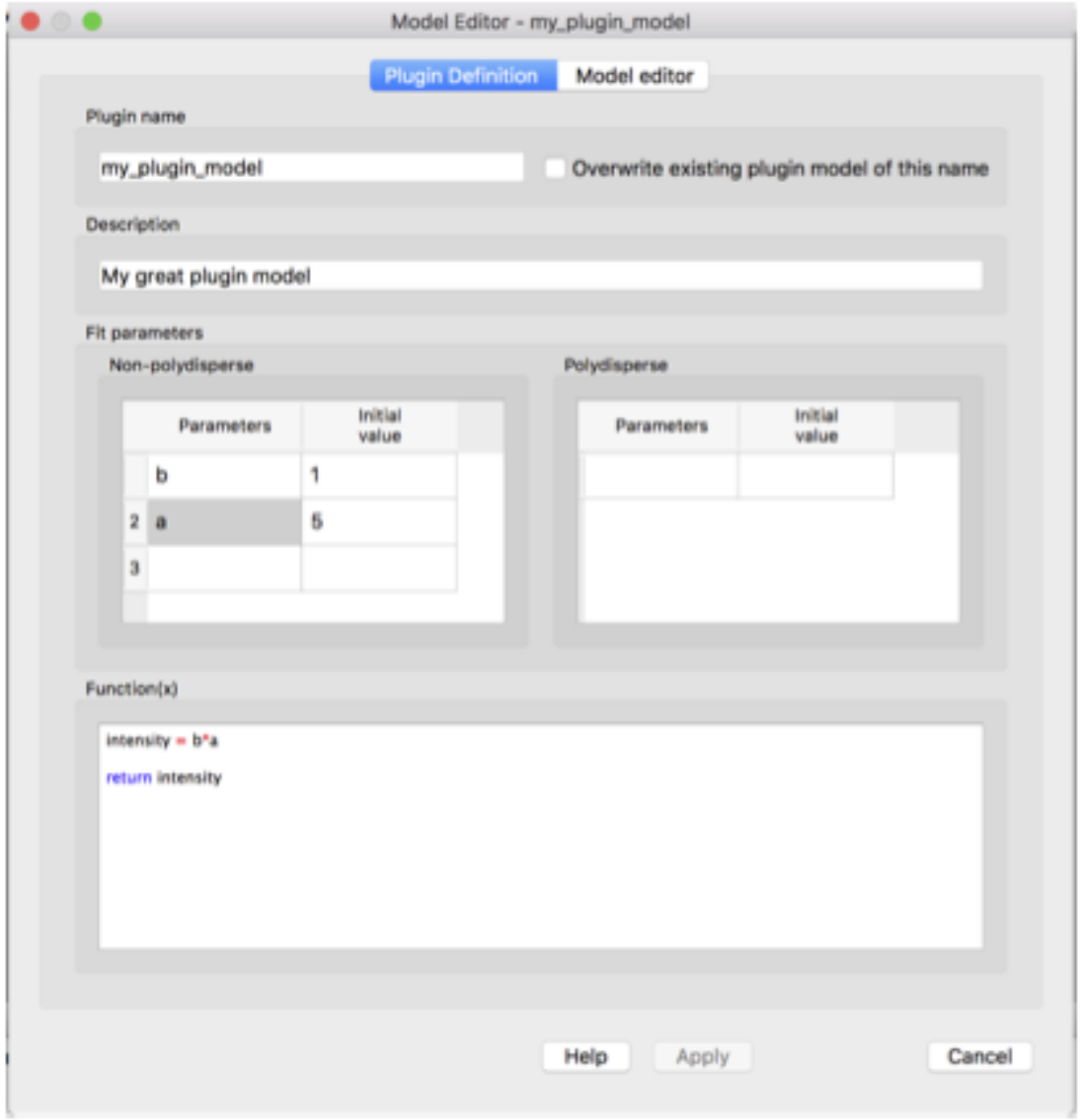


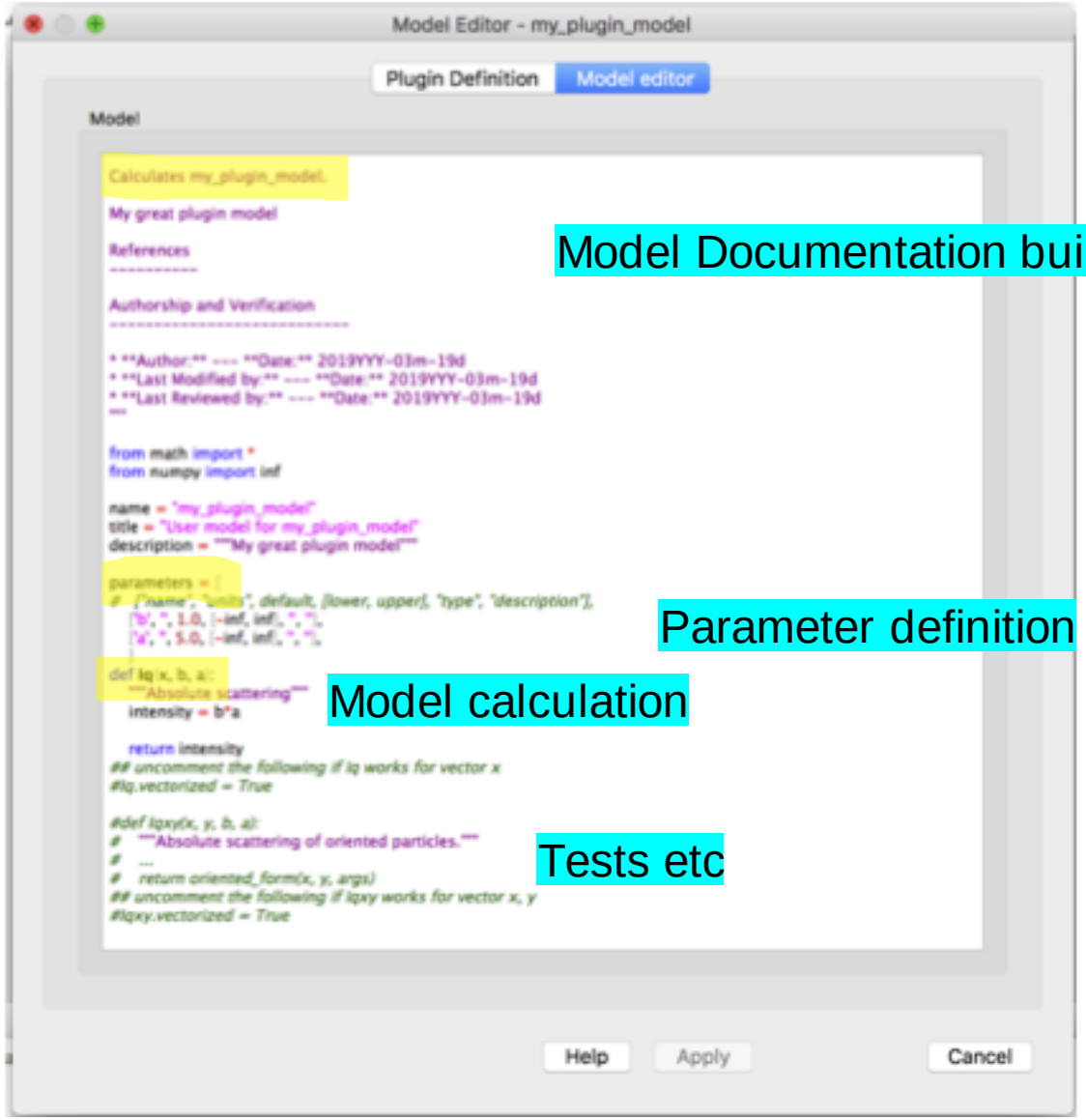




# Plugin Model Editor



# Plugin Model Editor: Python & C model files

cylinder.py

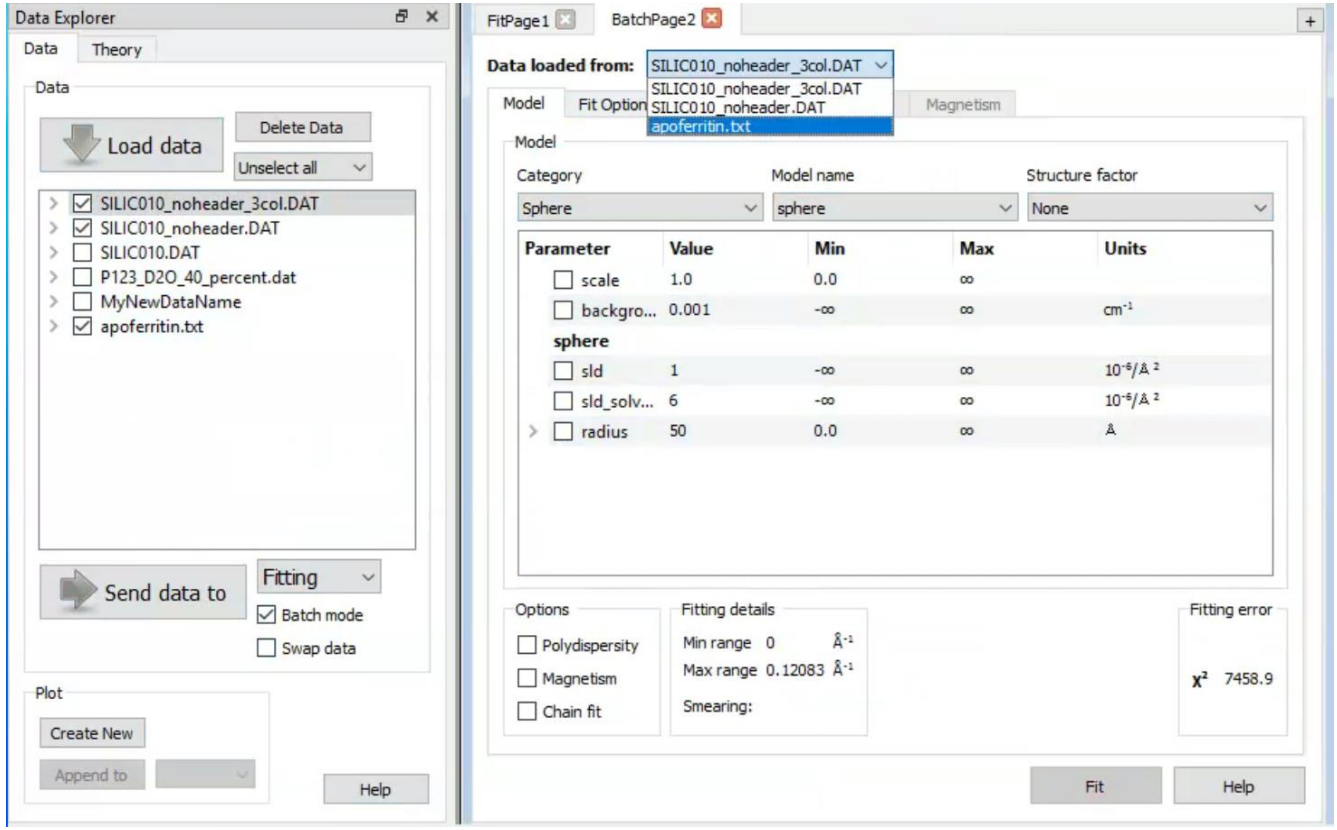
```python
import ...

name = "cylinder"
title = "Right circular cylinder with uniform scattering length density."
description = """
     f(q,alpha) = 2*(sld - sld_solvent)*V*sin(qLcos(alpha)/2))
                  /[qLcos(alpha)/2]*J1(qRsin(alpha))/[qRsin(alpha)]

           P(q,alpha)= scale/V*f(q,alpha)^(2)+background
           V: Volume of the cylinder
           R: Radius of the cylinder
           L: Length of the cylinder
           J1: The bessel function
           alpha: angle between the axis of the
           cylinder and the q-vector for 1D
           :the ouput is P(q)=scale/V*integral
           from pi/2 to zero of...
           f(q,alpha)^(2)*sin(alpha)*dalpha + background
"""
category = "shape:cylinder"

#            [ "name", "units", default, [lower, upper], "type", "description"],
parameters = [["sld", "1e-6/Ang^2", 4, [-inf, inf], "sld",
               "Cylinder scattering length density"],
              ["sld_solvent", "1e-6/Ang^2", 1, [-inf, inf], "sld",
               "Solvent scattering length density"],
              ["radius", "Ang", 20, [0, inf], "volume",
               "Cylinder radius"],
              ["length", "Ang", 400, [0, inf], "volume",
               "Cylinder length"],
              ["theta", "degrees", 60, [-360, 360], "orientation",
               "cylinder axis to beam angle"],
              ["phi", "degrees", 60, [-360, 360], "orientation",
               "rotation about beam"],
             ]

source = ["lib/polevl.c", "lib/sas_J1.c", "lib/gauss76.c", "cylinder.c"]

def ER(radius, length):
    """
        Return equivalent radius (ER)
    """
    ddd = 0.75 * radius * (2 * radius * length + (length + radius) * (length + pi * radius))
```

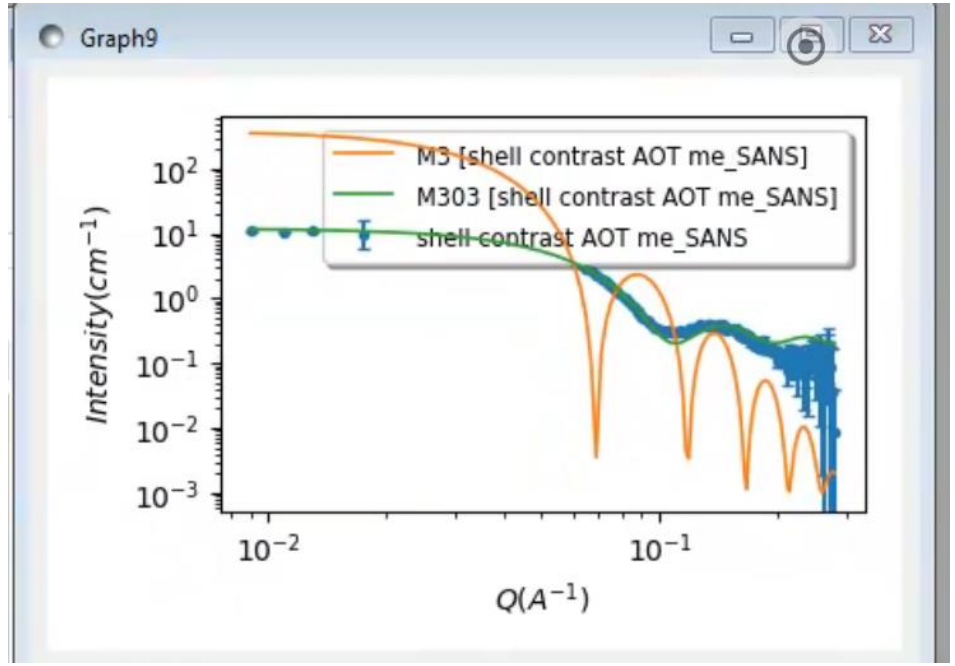
cylinder.c

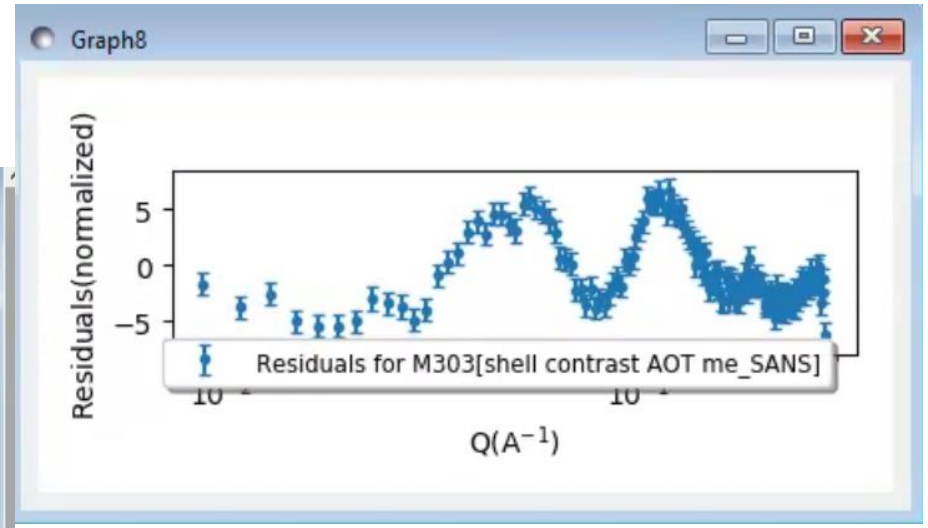
```c
#define INVALID(v) (v.radius<0 || v.length<0)

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);
}
```

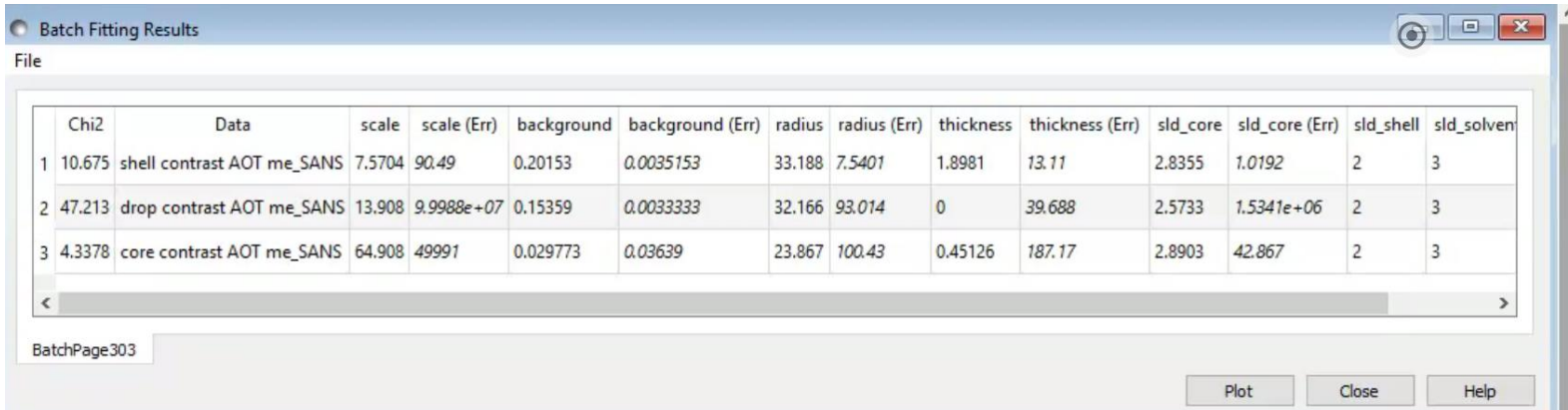
**Complex numbers not originally supported – a bit in C now**

# BATCH in Fitting

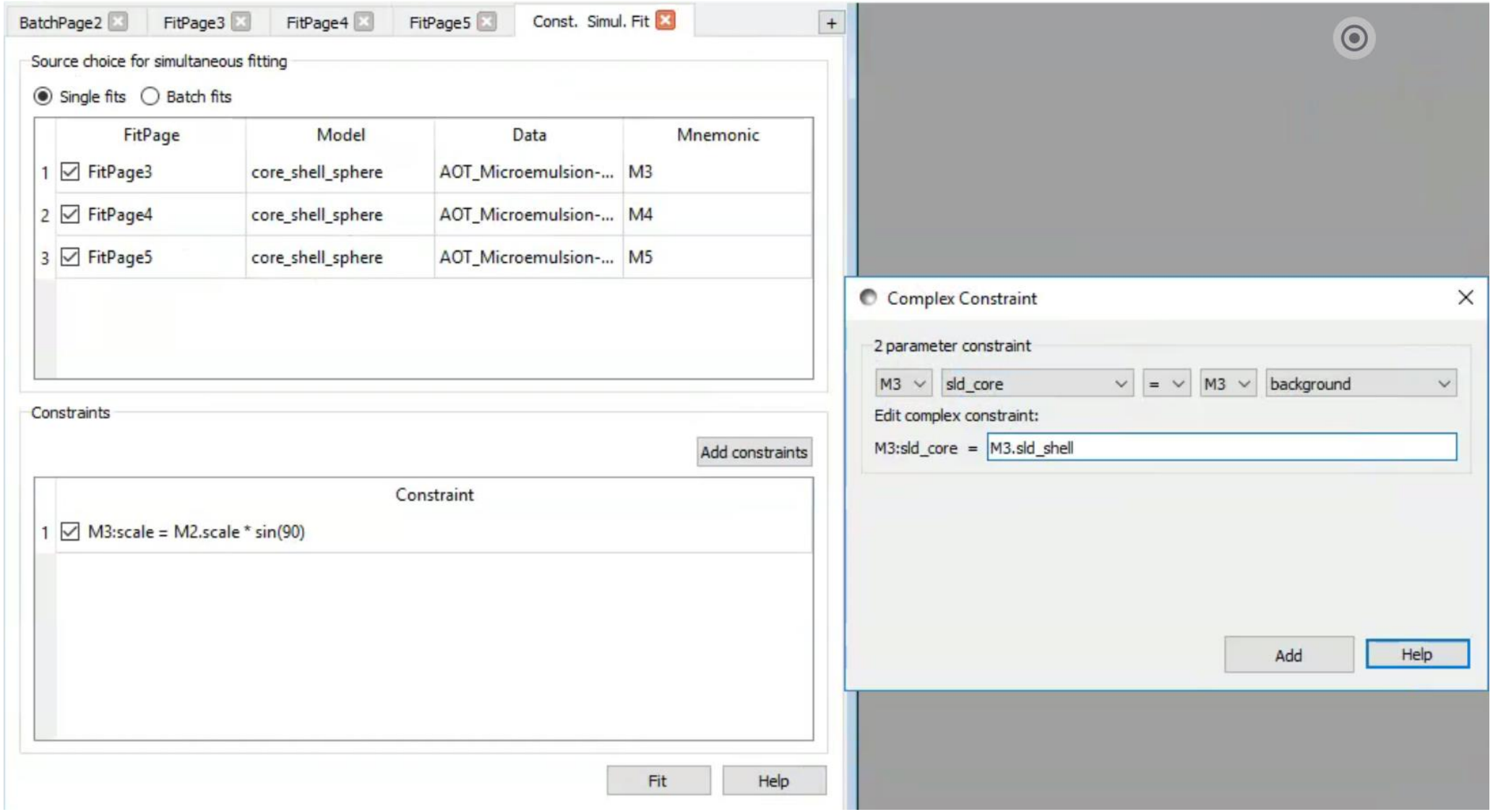


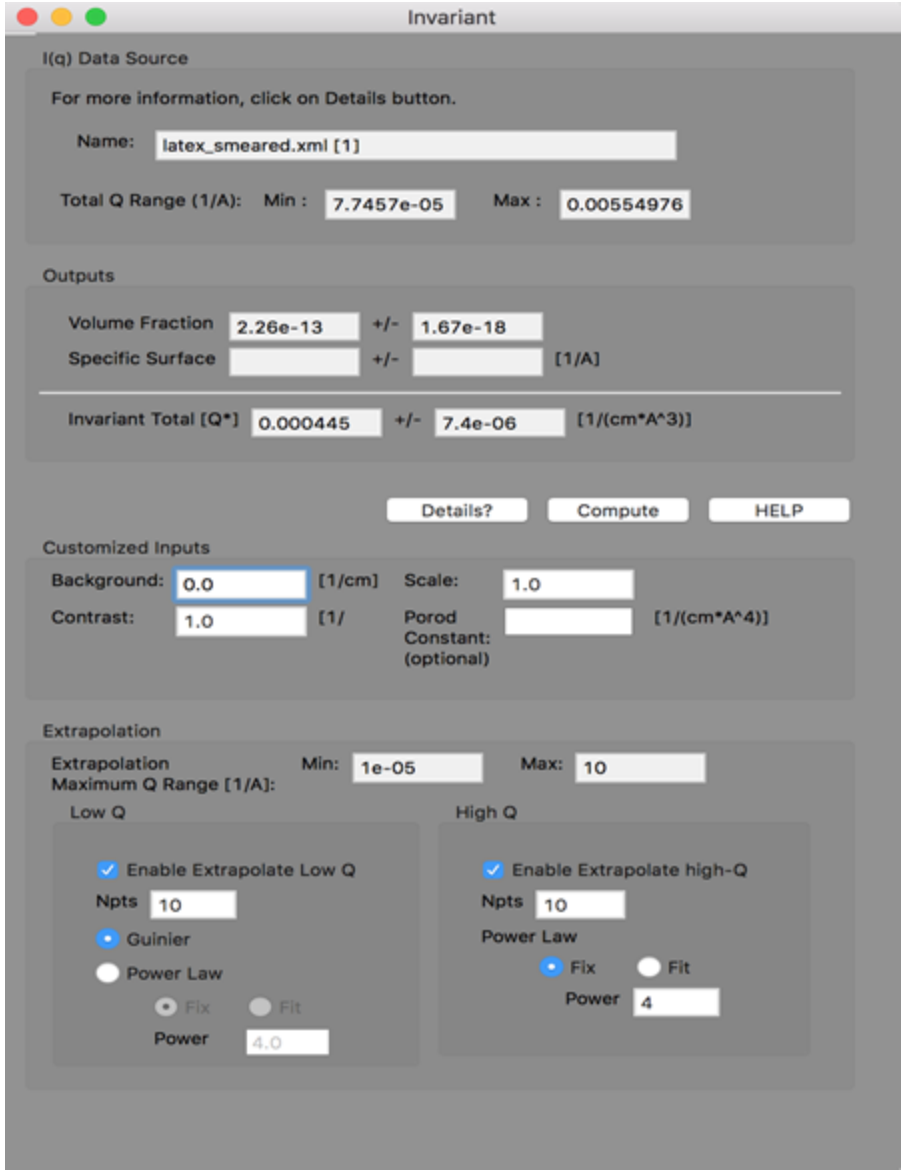



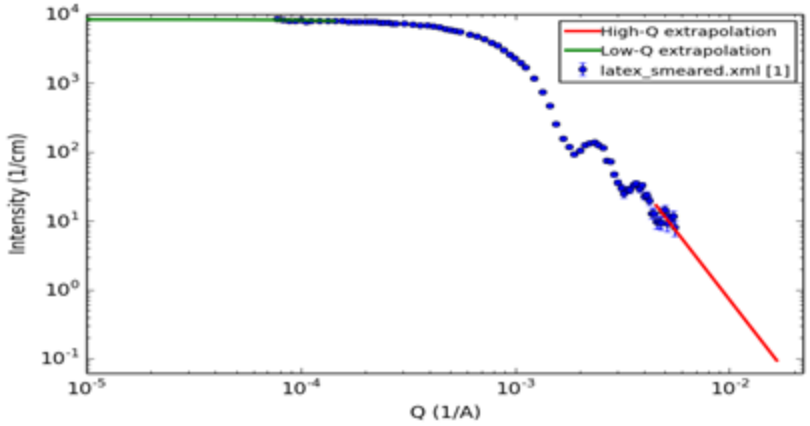



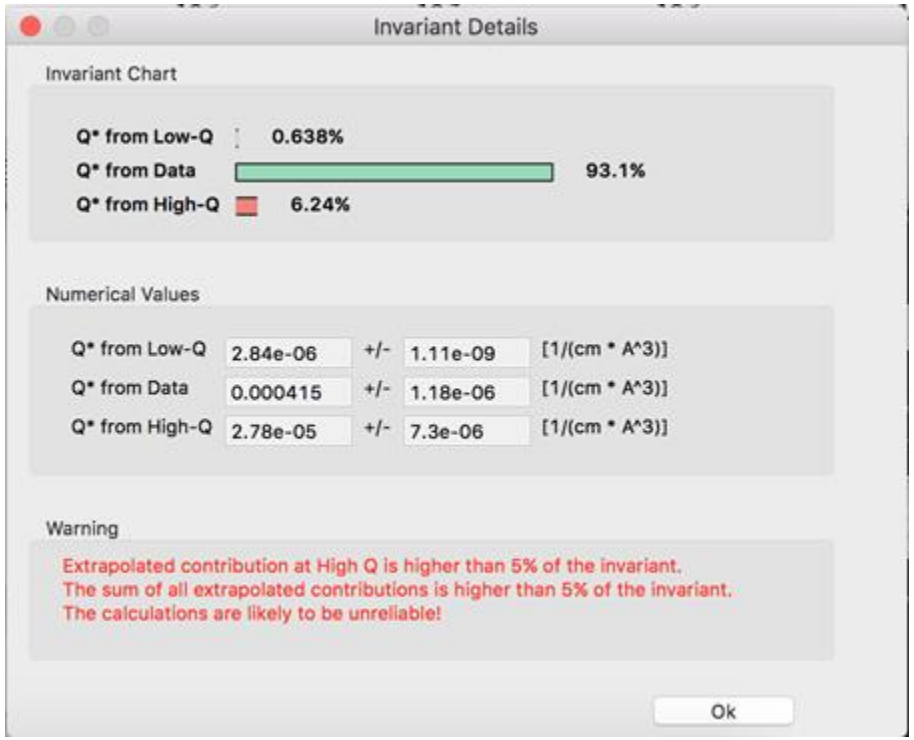
# Constrained Fitting



# Fitting Perspective: Invariant Calculation



# Fitting Perspective: P(r) Inversion

P(r) control panel

I(q) data source

Name: 34854_db_rear.xml

Estimate background level

Slit parameters

Height | Width | [A^(-1)]

Q range

Q min 0.015 | Q max 0.35 | [A^(-1)]

Parameters

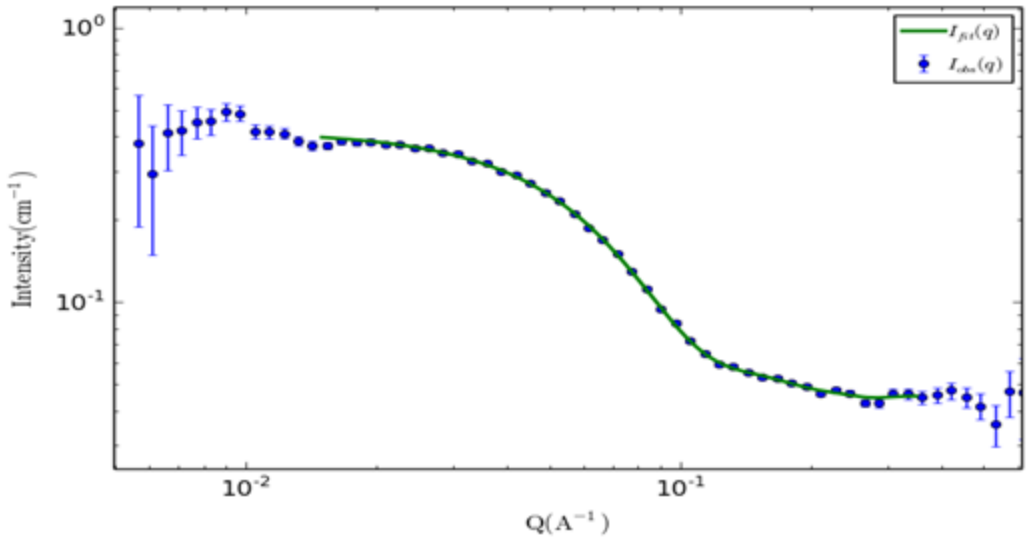
P(r) is found by fitting a set of base functions to I(Q). The minimization involves a regularization term to ensure a smooth P(r). The regularization constant gives the size of that term. The suggested value is the value above which the output P(r) will have only one peak.

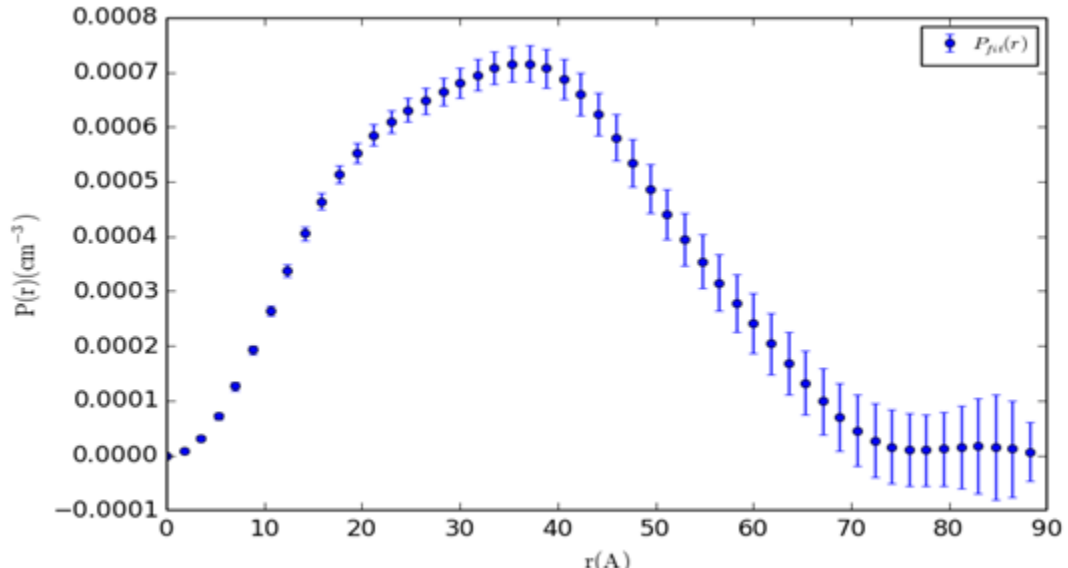
| | | Suggested value |
|---|---|---|
| Number of terms | 15 | 15 |
| Regularization constant | 4e+09 | 4e+09 |
| Max distance [A] | 90 | Explore |

Outputs

| | | |
|---|---|---|
| Rg | 27 | [A] |
| I(Q=0) | 0.37 | [A^(-1)] |
| Background | 0.046 | [A^(-1)] |
| Computation time | 0.0011 | secs |
| Chi2/dof | 46 | |
| Oscillations | 1.3 | |
| Positive fraction | 1 | |
| 1-sigma positive fraction | 0.99 | |

Compute HELP

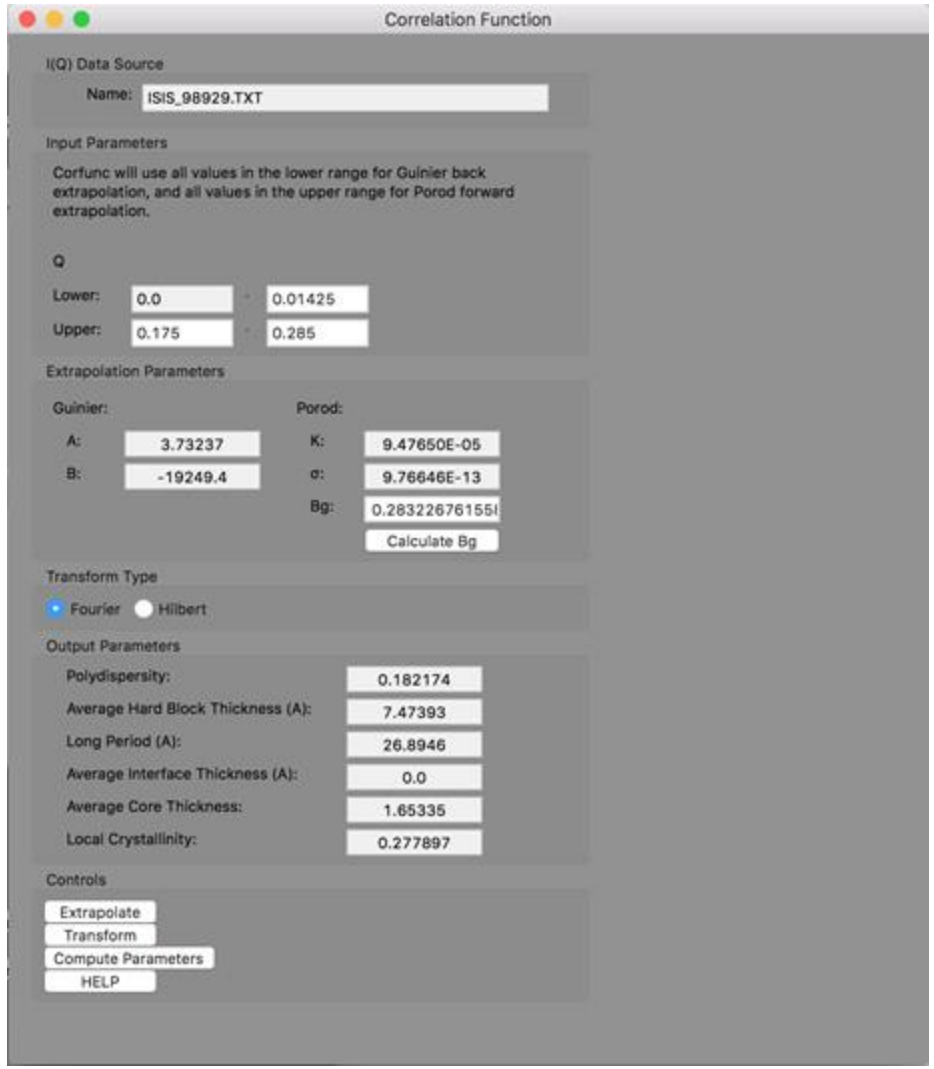


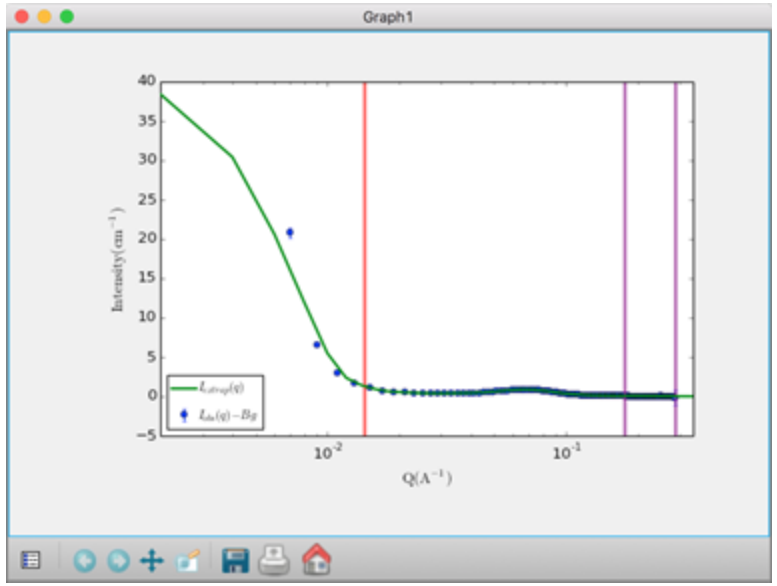



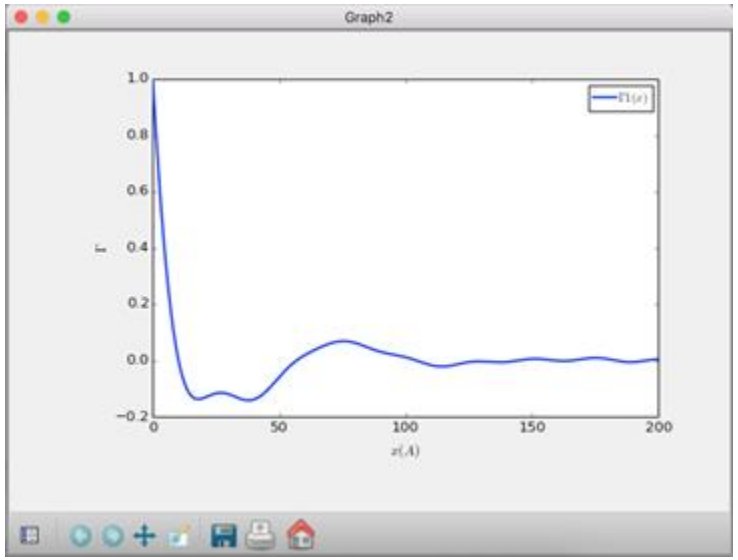
# Fitting Perspective: Correlation Function Analysis (new!)

CCP13 (Fiber Diffraction) legacy code (Fortran)

Implemented by an ISIS summer student)



# SESANS Analysis

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

Fit panel - Active Fitting Optimizer: Levenberg-Marquardt

FitPage1

I(q) Data Source

Name : sphere2micron.ses

Model [ M1 ]

Category: Sphere — Modify — Description — 1D Mode — Help

sphere — P(Q)*S(Q): None

Model Parameters

| Parameter | Value | | Error | Min | Max | [Units] |
|---|---|---|---|---|---|---|
| ✓ scale | 0.005323 | +/- | 5.2808e-0 | 0 | inf | |
| background | 0 | +/- | | -inf | inf | 1/cm |
| sld | 1 | +/- | | -inf | inf | 1e-6/Ang^2 |
| sld_solvent | 6 | +/- | | -inf | inf | 1e-6/Ang^2 |
| ✓ radius | 10383 | +/- | 114.45 | 0 | inf | Ang |

Polydispersity and Orientational Distribution

On — Off — ?

Fitting

Set Instrumental Smearing

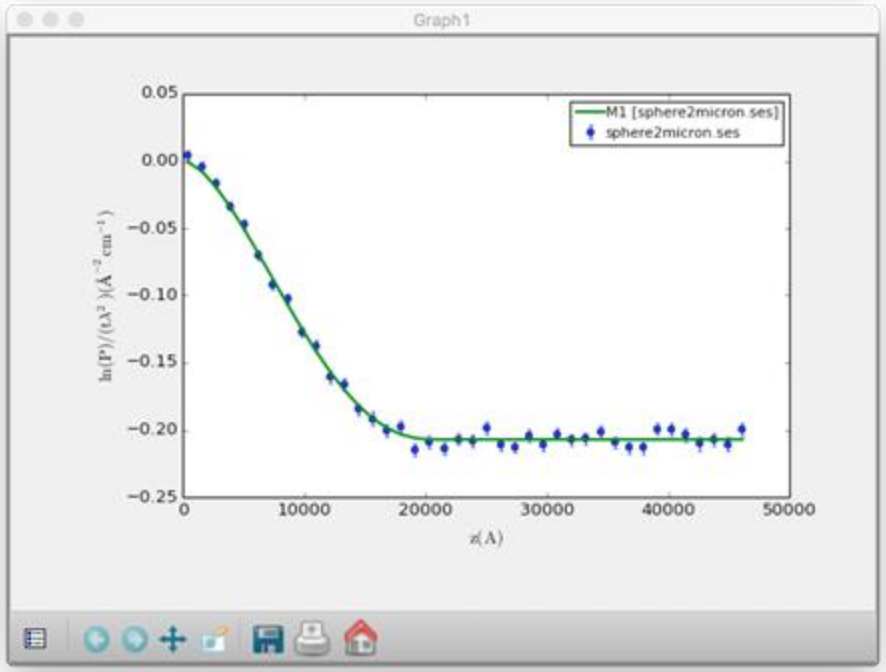
None — Use dQ Data — Custom Pinhole Smear — Custom Slit Smear — ?

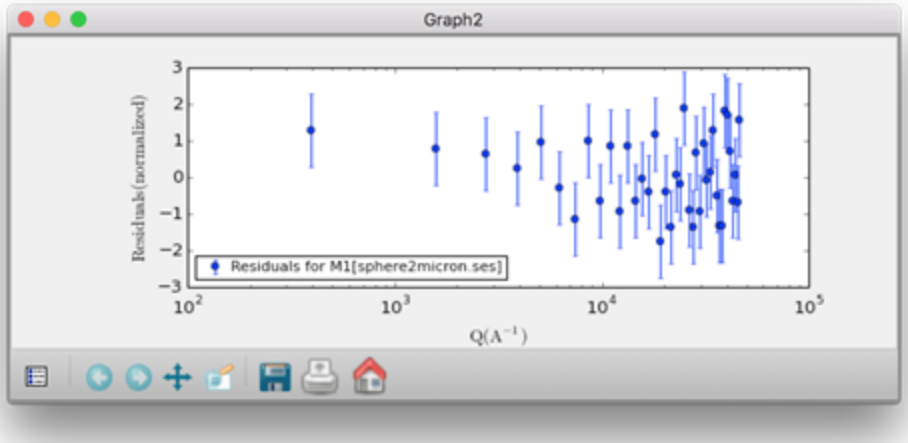
The dQ data is being used for smearing...

Type: Pinhole — 19.578 — dQ[%]: 2304.9

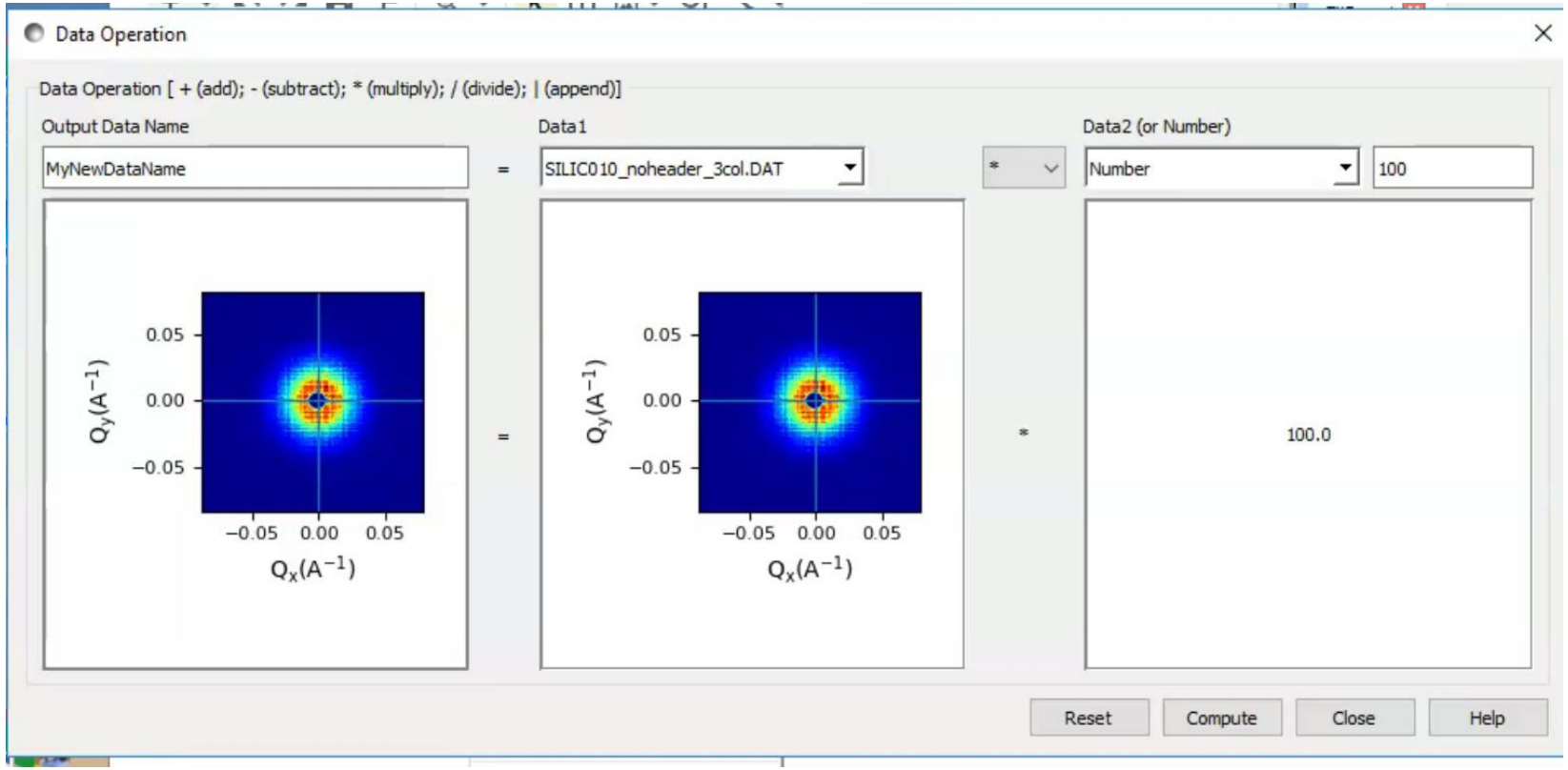
Set Weighting by Selecting dI Source

No Weighting — Use dI Data — Use |sqrt(I Data)| — Use |I Data|



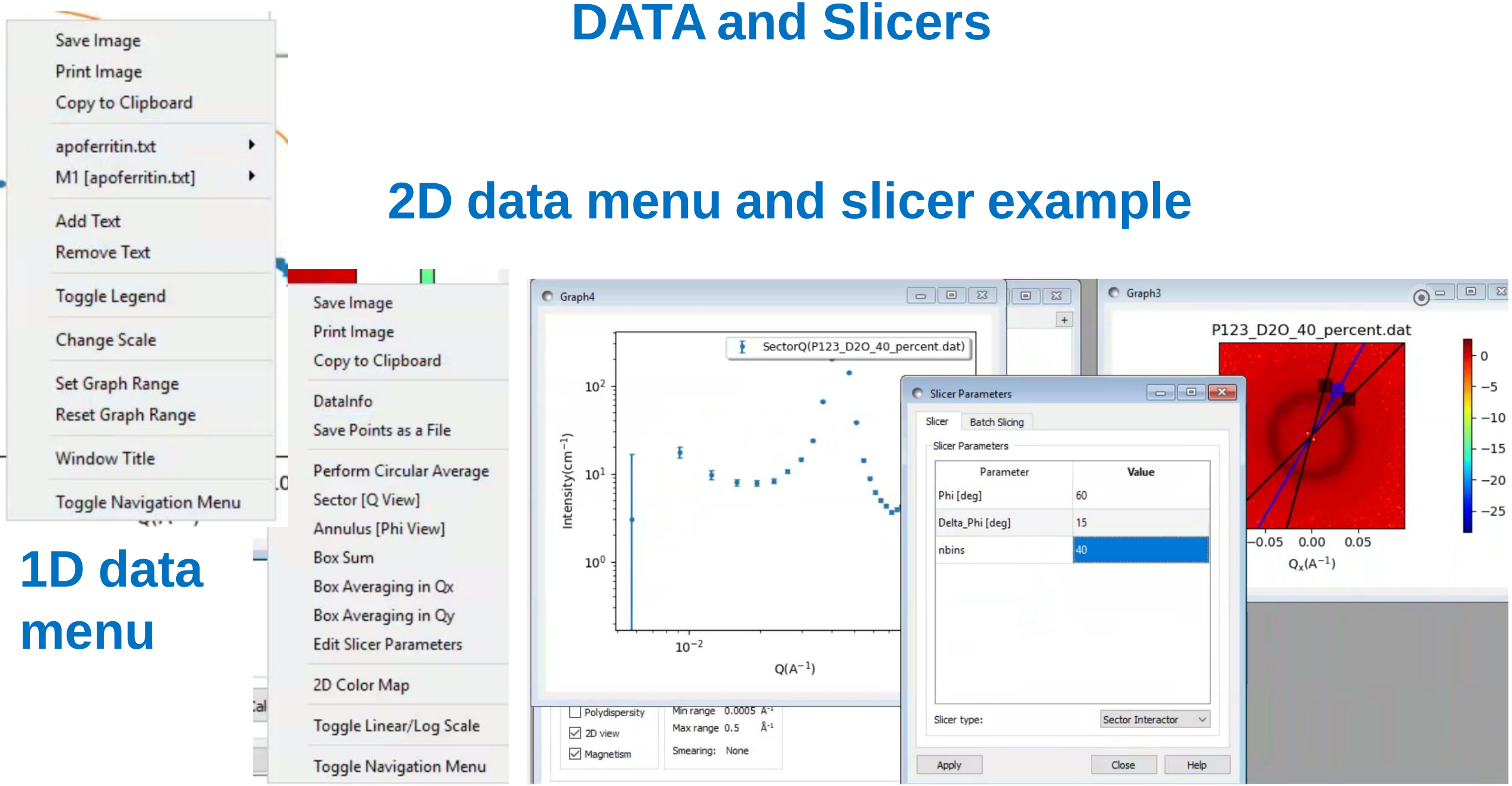
# Data Operations and 1D stuff




**PROBLEM: If using 2 data sets, ALL Q points must match (not interpolation etc)**

# DATA and Slicers

## 2D data menu and slicer example



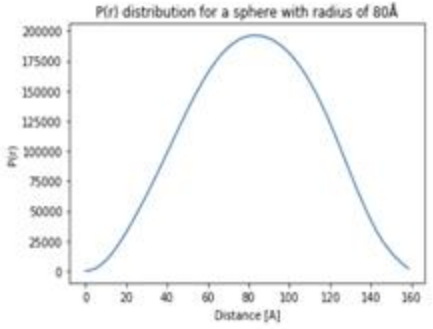

1D data menu

# Command Line and Jupyter Notebooks

## SasCalc example

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

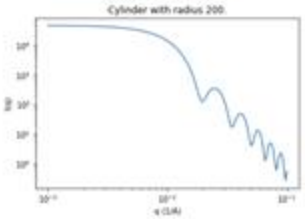
In [2]:
```
from sas.sascalc.dataloader.loader import Loader
from sas.sascalc.pr.invertor import Invertor
import matplotlib.pyplot as plt
import numpy as np

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 [A]")
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 strcuture 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.

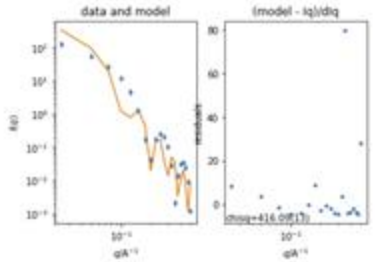
In [35]:
```
from numpy import logspace
from matplotlib import pyplot as plt
from sasmodels.core import load_model
from sasmodels.direct_model import call_kernel

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/A)')
plt.ylabel('I(q)')
plt.title('Cylinder with radius 200.')
plt.show()
```



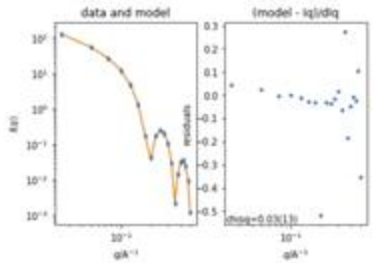

## Fitting model function to data using bumps

The model functions from sasmodels can be used to fit experimental data. This can be done using bumps, which simillar 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()
```

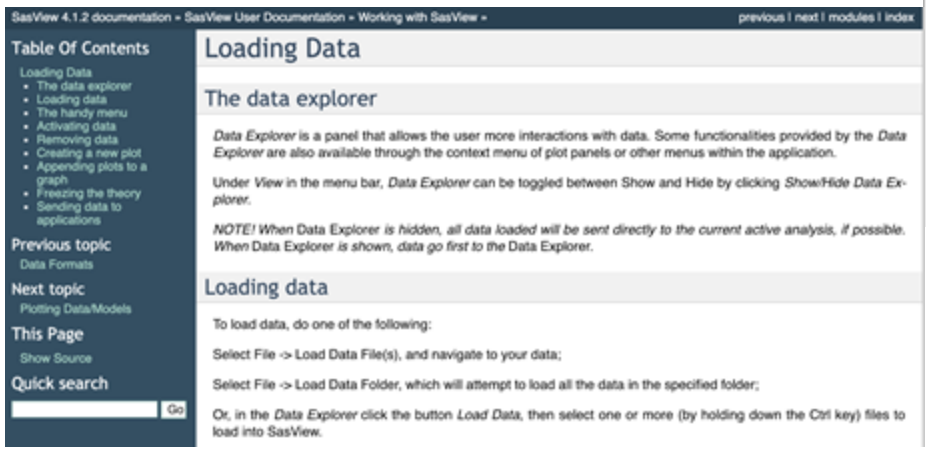



```
Final chisq 0.03(13)
length : 464.9(55)
radius : 19.977(64)
```

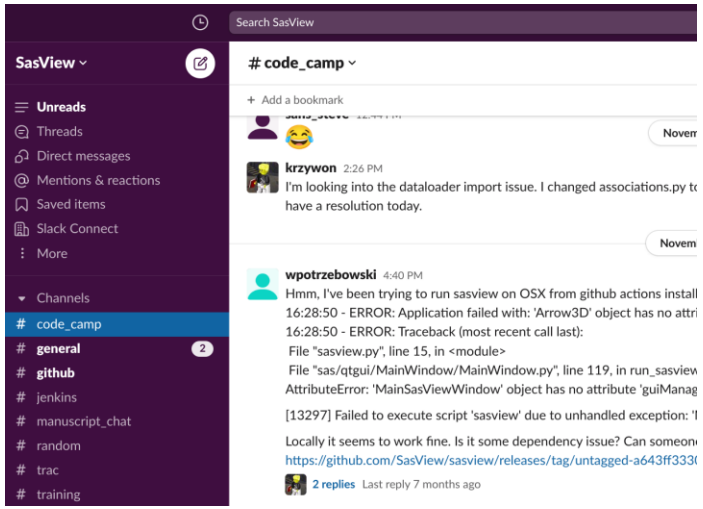


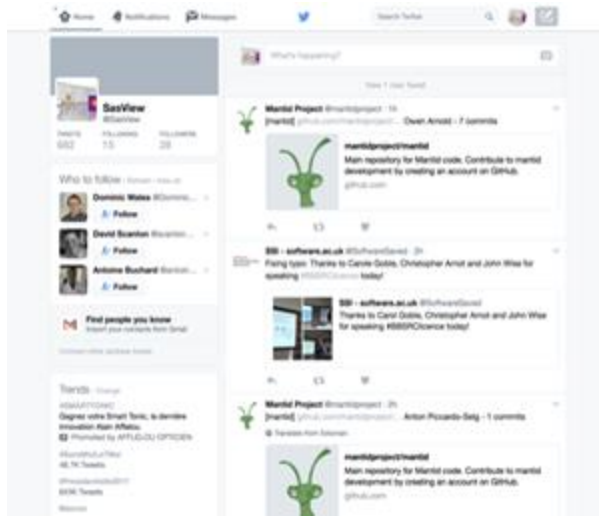
# Resources maintained and  Education & Outreach

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



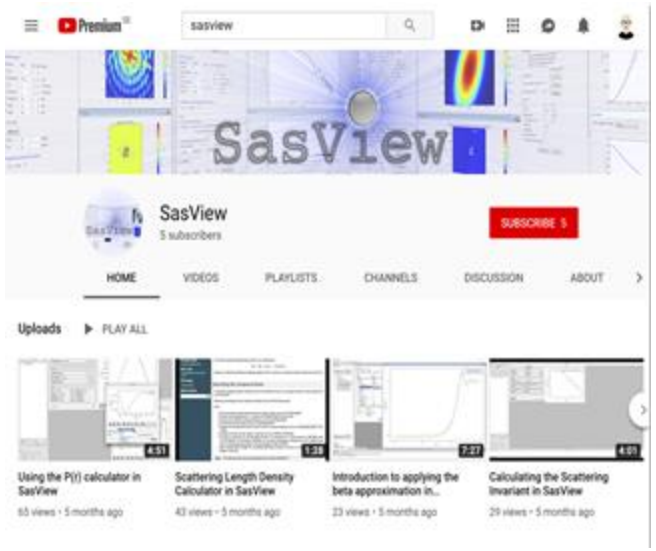


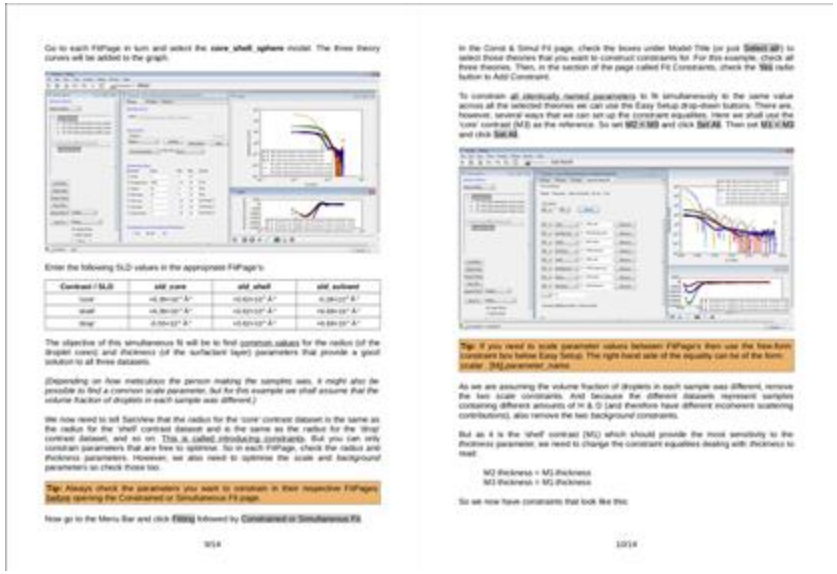


# Resources maintained and  Education & Outreach

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



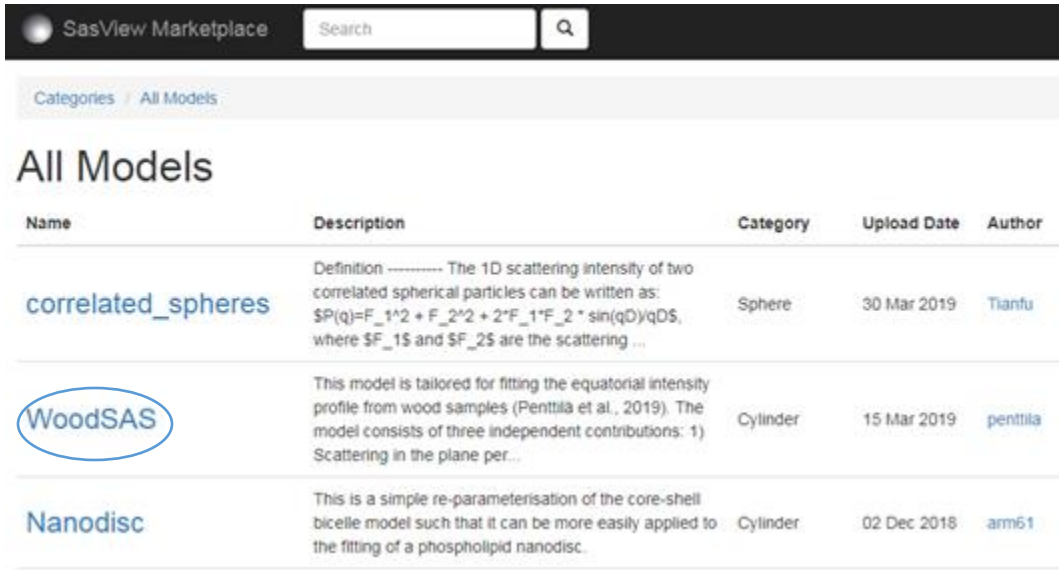



# Resources maintained and  Education & Outreach

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

*All the work of ISIS Sandwich Student Michael Oakley*


*Original put together by an SIS Summer Student Lewis O'Driscoll*



# 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 [1].

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 support, users of scattering facilities, in particular new users, have similar expectations of the software they use to operate the instrument and process and analyse their data.

**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(Q) and S(Q) components in a P*S analysis. Work will begin on integration of McSAS into SasView, primarily planning and design work.

The SasModels Marketplace will be updated to better support user needs and fix bugs in the deployment, including bringing the backend up to recent versions.

**Living document**

**Guides and tracks developers' work**
**Helps find candidate projects for funding.**

**Discussed and updated at each Code Camp.**

**Hope to update after these discussions!!**

**Think about different Workflows and how to implement**

https://github.com/SasView/documents

# 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 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
- *'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