SasView

An “open, collaborative, community development” platform for Small Angle Scattering Data Analysis
SasView

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

Focus on analytical approaches for this package
   .... Sorta

Whatever anybody puts into it
   .... Sorta
A little history ...

Heritage: NIST IGOR macros
SansView is DANSE project output
~ 8.5% of funds were for SANS
+ BUMPS ... see later

NIST Supported initial transition from NSF funding
Transition to Community project.

Move to GitHub
Rename to SasView

v3.0 released
v3.1 released
v4.0 released
v4.1 released

1st Code Camp at NIST April 2013
2nd Code Camp at ISIS April 2014
3rd Code Camp at ESS Feb 2015
4th Code Camp at TU Delft March 2016
5th Code Camp at ORNL Oct 2016
6th Code Camp at ILL/ESRF April 2017
7th Code Camp at DMSC October 2017
8th Code Camp at ESS Sept 2018

1st SasView User Meeting at SAS2018

9th Code Camp at ILL/ESRF March 2019

v4.2 released
v3.0 released
v4.0 released
v4.1 released
v5.0 released
v4.2.1 released
v5.0b2 released

v5.0b1 released
v5.0 released
v4.2 released
v4.2.1 released
v5.0b2 released
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
1D Analysis

Data management
- Common data formats supported, including NXCanas & 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
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
Fitting Algorithms

Uses bumps package from P. Kienzle
(also has DANSE origins)
Plugin Model Editor

**Plugin Definition**

- **Plugin name**: my_plugin_model
- **Description**: My great plugin model

**Fit parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

**Function(x)**

- intensity = b^a
- return intensity

**Model**

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

```python
from math import *
from numpy import *

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

def lnx(x, b, a):
    """Absolute scattering""
    intensity = b^a
    return intensity

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

def lnx(x, y, a):
    """Absolute scattering of oriented particles""
    #
    # return oriented_form(x, y, args)
    # uncomment the following if lnx works for vector x, y
    # npy.vectorized = True
```
Python & C model files

```python
name = "cylinder"

title = "Right circular cylinder with uniform scattering length density."

description = "Right circular cylinder with uniform scattering length density."

f(q, alpha) = 2*(sld - sld_solvent)+V*sin(q*cos(alpha)/2))
/q*cos(alpha)/2)=J1(q*sin(alpha))/[q*Rsin(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 ID
:the output is P(q)=scale*V*integral

from pi/2 to zero of...

f(q, alpha) = (2)*sin(alpha)*alpha + background

category = "shape:cylinder"

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

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

```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_Z31_x(qab=radius) * sas_sin_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[1i]*zm + zb;
        double sin_theta, cos_theta; // slits to hold sin/cos function output
        // theta (theta,phi) the projection of the cylinder on the detector plane
        SINCOS(theta, sin_theta, cos_theta);
        const double form = tf(q*sin_theta, q*cos_theta, radius, length);
        total += GAUSS_W[1i] * form * sin(theta);
    }
    // translate dx in [-1,1] to dx in [lower,upper]
    return total*zm;
}

static double
lq(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);
}
```
**Invariant Calculation**

### I(q) Data Source

For more information, click on Details button.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Q Range (1/A)</td>
<td>Min: 7.7457e-06 Max: 0.00554976</td>
</tr>
</tbody>
</table>

### Outputs

| Volume Fraction | 2.28e-13 +/- 1.67e-18 [1/A] |
| Specific Surface |                             |

| Invariant Total (Q*) | 0.000445 +/- 7.4e-06 [1/(cm*Å^3)] |

### Customized inputs

- **Backgrounds:** 0.0 [1/cm]
- **Contrast:** 1.0 [1]
- **Scale:** 1.0
- **Porod Constant:** (optional)

### Extrapolation

- **Extrapolation**
  - Min: 1e-05
  - Max: 10

- **Maximum Q Range [1/A]:**
  - Low Q
  - High Q

- **Enable Extrapolate Low Q:**
  - Npts: 10
  - Guinier
  - Power Law

- **Enable Extrapolate high-Q:**
  - Npts: 10
  - Power Law
  - Fix: 4.0

### Numerical Values

- Q' from Low-Q: 2.84e-06 +/- 1.11e-06 [1/(cm*Å^3)]
- Q' from Data: 0.000445 +/- 1.18e-06 [1/(cm*Å^3)]
- Q' from High-Q: 7.78e-06 +/- 7.3e-06 [1/(cm*Å^3)]

### Warning

Extrapolated contribution at High Q is higher than 5% of the invariant. The sum of all extrapolated contributions is higher than 5% of the invariant. The calculations are likely to be unreliable.
**P(r) Inversion**

I(q) data source
Name: 34854_db_rear.xml
- Estimate background level

Slit parameters
- Height
- Width [Å⁻¹]

Q range
- Q min: 0.015
- Q max: 0.35 [Å⁻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.

<table>
<thead>
<tr>
<th>Number of terms</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regularization constant</td>
<td>4.0e+09</td>
</tr>
<tr>
<td>Max distance [Å]</td>
<td>90</td>
</tr>
</tbody>
</table>

Suggested value
- 15
- 4.0e+09
- Explore

Outputs
- Rg [Å]
- I(Q=0) [Å⁻¹]
- Background [Å⁻¹]
- Computation time [secs]
- Chisq/dof
- Oscillations
- Positive fraction
- 1-sigma positive fraction

![Graphs showing P(r) and I(q) data with various parameters and outputs.](image)
Correlation Function Analysis (new!)

CCP13 (Fiber Diffraction) legacy code (Fortran)

(ISIS summer student)
SESANS Analysis

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

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

Tip: Always check the parameters before opening the Constrained or Fit Page.
Now go to the Menu Bar and click Run.
Resources, 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
Resources, 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

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
Resources, Education & Outreach

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

All the work of ISIS Summer Student Lewis O’Driscoll

Uses mySQL & Postgress

8 models contributed by 7 authors in 2 years
So …

Where is SasView Going?  
What will it do for me?
Scientific Software Development and The Cyberinfrastructure Revolution

- Never enough resources to achieve the vision we have
- No resources for long term maintenance and support.

CONCLUSION: This paradigm is broken!!!

FACTS OF LIFE:
- Resources are finite
- Needs are infinite
Analysis Software - Who’s Job is it Anyway?

Analysis is where the science is → the USER’S JOB

Scattering is an analysis tool and part of providing the tool should be the necessary software → the FACILITY’S JOB

Data sat on disk is useless to EVERYBODY

We need to work together!
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

---

http://github.com/SasView
http://www.sasview.org
How We Work

Open, Collaborative, Community Development

DOI for each release

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

http://github.com/SasView
http://www.sasview.org
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.

http://www.sasview.org  http://github.com/SasView
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
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

SasView

SasGui
- SasGUI
  - guiframe
  - perspectives
  - plottools

SasCalc
- SasCalc
  - calculator
  - data_util
  - dataio
  - fit
  - invariant
  - pr
  - simulation

SasModels
- Scattering models
- OpenCL support
- Available from pypi

Bumps
- Optimizer with Bayesian uncertainty analysis
- Available from pypi
- P. Kienzle
**Jupyter Notebooks**

**SasCalc example**

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

```python
from sas.sascalc.data_loader import Loader
from sas.sascalc.lab.inverter import Inverter
import matplotlib.pyplot as plt
import numpy as np

loader = Loader()
y_data = test_data[0].y
x_data = test_data[0].x

pr = Inverter()
pr.y = y_data
pr.x = x_data
pr.err = x_data

pr.alpha = 2.6e-5
pr.d_max = 180

#func - number of base functions to use.
out, cov = pr.invert(nfunc=13)
pr_value = []
err_value = []
for r_i in r:
    (Value, err) = pr.pr_err(out, cov, x_i)
    pr_value.append(Value)
    err_value.append(err)

plt.plot(r, pr_value)
plt.xlabel('Distances [A]')
plt.ylabel('$P(r)$')
plt.title('P(r) distribution for a sphere with radius of 80A')
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.

```python
from sasmodels import load_model
from sasmodels.plot.plot import plot
from sasmodels.core import load_model
from sasmodels.direct_model import call_kernel

model = load_model('cylinder')
q = np.linspace(0.3, 1, 200)
kq = call_kernel(kernel, q)
plt.plot(q, kq)
plt.xlabel('$q (\text{\AA}^{-1})$')
plt.ylabel('$k(q)$')
plt.title('Cylinder with radius 200.')
plt.show()

plt.plot(q, kq)
plt.xlabel('Distance [\text{\AA}]')
plt.ylabel('Intensity')
plt.title('P(r) distribution for a sphere with radius of 80\text{\AA}')
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 similar to sasmodels is a separate package and needs to be installed in your path.

```python
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 numpy as np

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,
            std=4.0,
            std_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(prob 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)
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.

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 the expectation of 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(O) and S(O) components in a P*S design

https://github.com/SasView/documents
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 Done – in v5.0
- 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\(^7\) Work done, but not shipping by default. https://github.com/SasView/sasfit-models
  b. Scatter\(^8\) (Förster - crystalline materials models primarily) In discussions with BornAgain team
- Project infrastructure cleanup:
  a. ticket review/cull given 5.0 release Done
  b. possible move to GitHub issues. Done
- Release
  a. 5.0 alpha (late 2018) Done
  b. 5.0 beta (early 2019) Done
  c. 5.0 (mid 2019) Done
  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

Official Releases available for Windows, Mac, and Debian Linux

**Models**
New models
New model package (sasmodels)
*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 – 5.0 current

Parallel release of 4.x and 5.x until 5 series is stable. 5.0 release out now and in use!

www.sasview.org

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

Sasmodels Enhancements
Return F(q) from models
Beta approximation
Coherent sums

Constraints refactor
Multi-GPU support
Inclusion of SasFit models

Integration of McSAS
Integration of PyPrism?
Implementation of key models from Scatter

Documentation
Tutorials – written, interactive & video
Manual

And much more!
See Roadmap and Issues on GitHub
Status of contributor community

“Management” Team
- Paul Butler (NIST)
- Mathieu Doucet (ORNL)
- Andrew Jackson (ESS)
- Steve King (STFC)

- 9 facilities
- 40 contributors on github (does not count original team)
- about 15 “active” at any one time

New people … now getting student interest
- Dominique Dressen (Sabrina Disch student, Köln Uni)
- Rachel Ford (Julia Kornfield student, Caltech)
- CARR (Tianfu and Dongfeng)
- CSNS (strong interest – couldn’t come for last camp due to visa issues)
- NSLSII → conda forge and deploy on their instruments
- ALS using sasmodels
- Users submitting models to marketplace
Saview Bootcamp? Training courses?
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

Day One = Using SasView
  • morning = overview lectures
  • Afternoon = hands-on/tutorial

Day Two = Write your own model - hands-on

Day Three = using sasview via scripting
  • Morning = Python tutorial
  • Afternoon = Intro to scripting with Jupyter Notebook and using to script sasview

Day Four = Contribute - preparation for code camp
  • includes contributing to tutorials, documentation, checking and fixing math, adding tests, reporting using issues, GUI framework code, marketplace database, etc.

https://github.com/SasView/documents/blob/master/Training/SasView_Boot_Camp/syllabus.md

http://www.sasview.org  http://github.com/SasView
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

Next SasView Food Code Camp (10th code camp!) will be held next year at Caltech – likely March/April
Questions?
Roadmap Late 2019 to mid 2020

- Begin model fitting refactoring work to allow custom re-parameterization of models, allow reading in an array representing either PQ or SQ for P*S fits, fitting oriented model to 1D cuts including revisiting orientation definitions etc. *Discussed at last code camp*
- Complete architecture manual
- Begin work on refactoring constrained/simultaneous fits.
- **Begin work on adding custom workflows identified as highest priority**
- Work to update tutorials to support 5.x
- **Begin work on advanced model fitting tutorial**
- Usual bug fixes and other minor improvements as time and interest permit
- Integration of McSAS
- Begin work on generic O-Z solver
- Inclusion of PRISM functionality
- **Begin work to refactor/improve generic scattering calculator**
- Improvements to custom model editors including features from compare.py
- Support for multi-GPU, multi-CPU and CPU/GPU computation

https://github.com/SasView/documents
Roadmap mid 2020 to mid 2024

- Refactor Simultaneous/Constrained fitting - significant changes in 5.0
- New Workflows
- Web UI (and Phone App)
- Headless - essentially done in 5.1?
- Intelligent limits/help ⇒ “AI”?
- Add support for ASAXS
- Enable transparently running computational code remotely from within local GUI - dependent on headless

https://github.com/SasView/documents