The Generic Scattering Calculator

Motivation

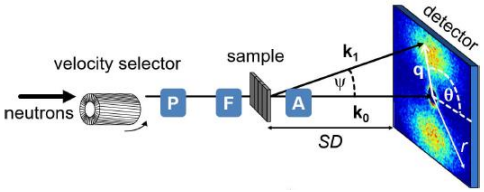
*Presentation information last updated 14/9/21 – alterations to the interface/functionality of Sasview after this date not included

Many different processes result in samples which may be analysed by small angle neutron scattering, and these are often modelled computationally, such as molecular dynamics simulations whose output can produce nuclear scattering patterns, or micromagnetic simulations which can produce magnetisation fields and hence magnetic scattering.

It is highly desirable to be able to predict what the scattering pattern from a sample. Especially when magnetisation is involved the results can be quite complicated, and Sasview contains a dedicated tool to predicting such patterns, given a known structure, created manually, programmatically, or as the output of a simulation.

In this presentation we will first take a brief look at the theory behind magnetic small angle neutron scattering, and then look at how we can use the generic scattering calculator to predict scattering intensity patterns.

MAGNETIC SANS



Magnetic small-angle neutron scattering: DOI: 10.1103/RevModPhys.91.015004

P: Polariser
F: Spin flipper
A: Analyser

- We can set the input polarisation direction of the neutrons
- We can choose to only see the scattering due to one output polarisation of neutrons
- This gives us 4 different intensity profiles (cross sections)

POLARIS – with analyser

Spin-flip	$\frac{d\Sigma^{+-}}{d\Omega}$	$\frac{d\Sigma^{-+}}{d\Omega}$
Non-spin-flip	$\frac{d\Sigma^{++}}{d\Omega}$	$\frac{d\Sigma^{--}}{d\Omega}$

SANSPOL – without analyser

$$\frac{d\Sigma^{+}}{d\Omega} = \frac{d\Sigma^{+-}}{d\Omega} + \frac{d\Sigma^{++}}{d\Omega}$$

$$\frac{d\Sigma^{-}}{d\Omega} = \frac{d\Sigma^{-+}}{d\Omega} + \frac{d\Sigma^{--}}{d\Omega}$$

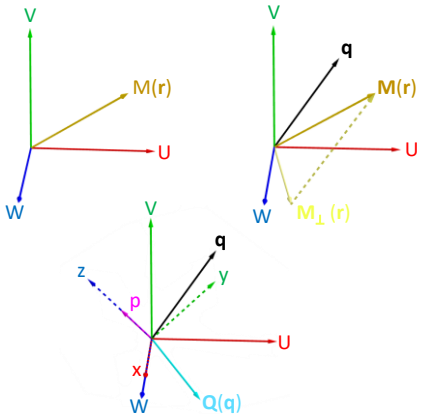
SANS - unpolarised

$$\frac{d\Sigma}{d\Omega} = \frac{d\Sigma^{+}}{d\Omega} + \frac{d\Sigma^{-}}{d\Omega}$$

In order to carry out magnetic small angle neutron scattering we can add 3 components to the standard beamline: a polariser, spin flipper and analyser. These allow us to choose the polarisation of neutrons entering the sample, and filter out only these we wish to analyse after the sample.
We will only consider the case when these are aligned in the same axis

Choosing the input and output polarisation allows us to consider four intensity patterns (called cross sections), as well as various combinations of these. Sometimes particular features only show up in certain cross sections (such as in the magnetic sphere example later) because interactions with magnetised samples can flip the polarisation of the neutrons.

MAGNETIC SANS



Fourier Transform

$\mathbf{M(r) \rightarrow \tilde{M}(q)}$

Halpern-Johnson vector

$\mathbf{Q} = \hat{\mathbf{q}}\left(\hat{\mathbf{q}} \cdot \tilde{\mathbf{M}}(\mathbf{q})\right) - \tilde{\mathbf{M}}(\mathbf{q})$

$b_H = 2.70 \times 10^{-15} m$

$$\frac{d\Sigma^{\pm\pm}}{d\Omega} = \frac{8\pi^3}{V}\left(\left|\tilde{N}\right|^2 \pm b_H\left(\tilde{N}Q_z^* + \tilde{N}^*Q_z\right) + b_H^2|Q_z|^2\right)$$

$$\frac{d\Sigma^{\pm\mp}}{d\Omega} = \frac{8\pi^3}{V} b_H^2(|Q_x|^2 + |Q_y|^2 \pm i\,(Q_xQ_y^* - Q_x^*Q_y))$$

1. For further details see:
Magnetic small-angle neutron scattering
Sebastian Mühlbauer, Dirk Honecker, Élio A. Périgo, Frank Bergner, Sabrina Disch, André Heinemann, Sergey Erokhin, Dmitry Berkov, Chris Leighton, Morten Ring Eskildsen, and Andreas Michels
DOI: 10.1103/RevModPhys.91.015004

2. Sign convention and choice of coordinate axes vary across the literature, Sasview uses a system explained below

Much as the nuclear scattering length density controls nuclear scattering, the Magnetisation of the sample controls the magnetic scattering. In fact by multiplying the magnetisation by a fixed constant b_H we obtain a comparable quantity known as the magnetic scattering length density.

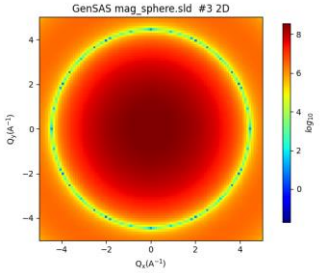
In fact only the component of this orthogonal to the neutron wavevector is important, and the Fourier transform of this is known as the Halpern-Johnson vector.

By defining a suitable coordinate system, equations can be derived for many circumstances, although the two most common are where the polarisation direction is either perpendicular or parallel to the beamline. For example when the polarisation vector is perpendicular to the beamline we can define a cartesian coordinate system with x along the beamline, and z along the polarisation vector. This then gives the equations shown for each cross section in terms of the components of the Halpern-Johnson vector in the xyz coordinate system.

The generic scattering calculator allows the polarisation vector to have any orientation relative to the beamline.

MAGNETIC SANS

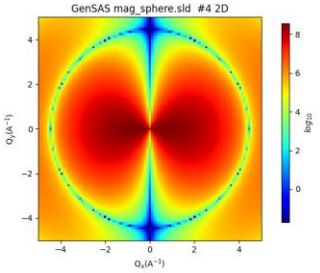
In the scattering pattern this can introduce angular anisotropies:



A single sphere with a constant nuclear SLD

$I(\boldsymbol{q}) \propto F(r)$

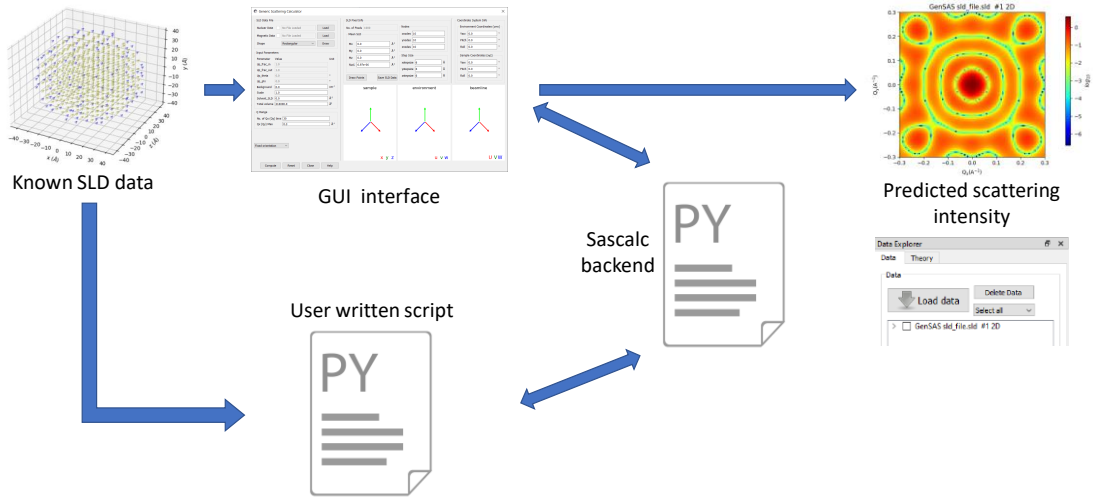
A single sphere with a constant nuclear SLD and equal magnetic SLD in the x direction, with the polariser and analyser in the x direction



$I(\boldsymbol{q}) \propto F(r) \times \cos^4\theta$

As an example consider a sphere. With purely nuclear scattering we get the expected circularly symmetric pattern. If we introduce a symmetry breaking magnetisation, aligned perpendicularly to the beam – we also gain an angular anisotropy in the scattering pattern.

WHAT IS THE GSC?



If we have a sample, and we know (or can estimate) it's scattering length densities, we should be able to calculate the expected scattering intensity pattern.

The generic scattering calculator is a GUI interface in Sasview (located in the tools menu) which allows these intensities to be easily calculated and displayed.

It uses the python backend to perform the scattering calculations.

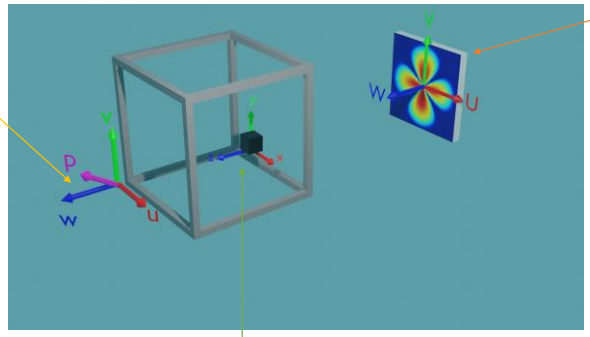
The output is displayed both as an image, and moved to the Sasview data explorer where it can be compared with standard models or other data.

For more complex operations user written python scripts can interface with the backend to produce results the GUI could not (an example of this to calculate orientational averages is shown later)

COORDINATE SYSTEMS

Environment coordinates
Describe the cryostat/polariser/ analyser, etc.

Relative to the beamline coordinates



Beamline coordinates remain fixed

Sample coordinates
Describe the real space positions and magnetic SLDs of the sample

Relative to environment coordinates

The calculator itself uses three different coordinate systems, although for simple cases these can be ignored, as they default to being aligned.

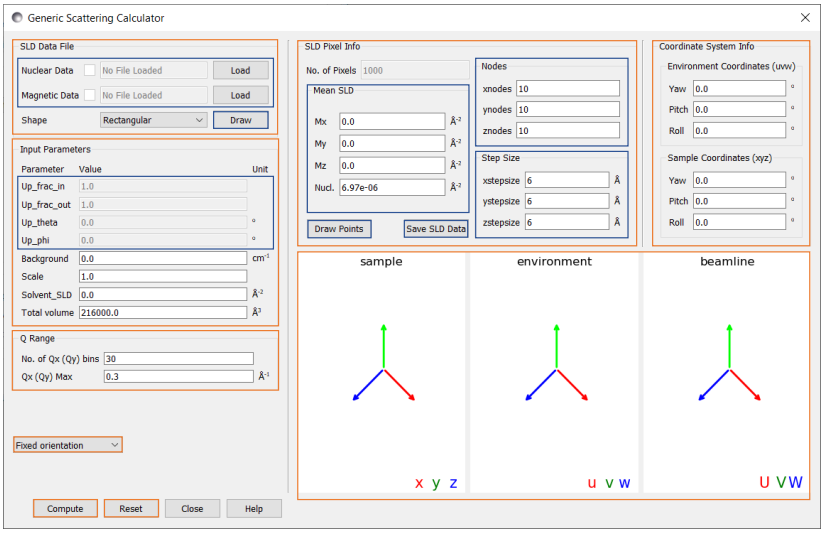
The beamline coordinates (upper case) {U, V, W} are static and define the Qx-Qy plane of the beamline.

The environment coordinates (lower case) {u, v, w} describe the orientation of the sample environment, and it is within these coordinates that the polarisation vector is set.

The sample coordinates {x, y, z} describe the rotation of the sample to the environment. These are very useful if the description of a sample is at a different orientation than that desired (see cylinder example later).

These coordinate systems also makes it very easy to perform rocking scans, by rotating the relevant coordinate system in the GUI

LAYOUT OF THE GUI



The scattering calculator has several panels which control different aspects of the calculator.

The first panel describes the location and type of the data files. Nuclear and magnetic scattering length densities can be loaded from files into the calculator, and enabled or disabled by the checkboxes. The draw button creates a visual representation of the data for the user.

The second panel describes the sample itself. The number of pixels or elements is given, and then the average values of the scattering length densities. If no datafile has been enabled for a scattering length density then a constant value can be given by the user which is applied to all pixels or elements.

For data in a regular grid the number of nodes describes the number of pixels in each axis, and the stepsize describes the spacing between each pixel. These values can be manually entered if no datafiles have been loaded – providing a very basic set of default data – useful for testing purposes. The draw points button draws the data points without any magnetic arrows – which can slow down the standard draw functionality for larger datasets. The save sld data button allows the current data to be saved into an sld file (a Sasview format) provided it is a rectangular grid.

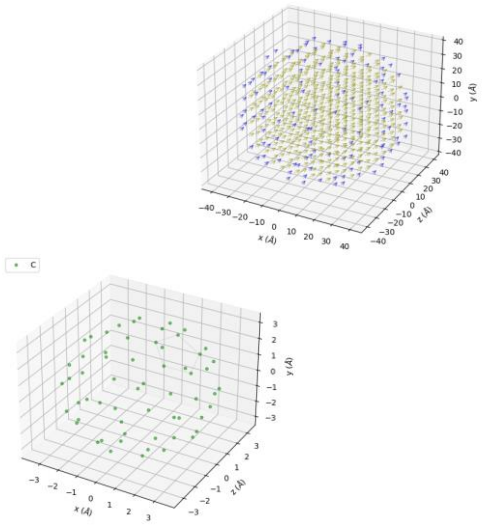
The coordinate systems information allows the user to view and edit the relative orientations of each of the coordinate systems, using yaw, pitch and roll angles. This is useful when a datafile has been created in a specific orientation, and the scattering pattern may be desired at another rotation. The GUI also shows an interactive 3D view of the coordinate systems for reference. The current version shown here contains an implementation in Matplotlib, although this is subject to change in the future.

The next panel describes parameters of the sample environment. The total volume is included here because it can be altered by the user to provide a correction factor for the intensity. The four parameters describing the magnetic beamline setup only become editable when a magnetic scattering length density is present – either due to a magnetic datafile or a manually entered value. In Sasview the extent of polarisation is described by values which give the fraction of neutrons polarised ‘up’. What has been notated as the ++ cross-section is given here as up_frac_in and up_frac_out both equal to 0, which will be seen in the examples later.

The Q range panel allows the detector settings to be entered – describing the range of Q values and the resolution of the detector. For some types of data – non-magnetic data on a regular grid, Sasview can also calculate the orientational average using the Debye formula, which can be set here.

The compute button is used to begin a calculation, and the reset button resets the interface to its initial settings.

DATA TYPES I - grid type data



File Types

- SLD files
  - A custom format for Sasview
  - Stores the positions and nuclear/magnetic SLDs, as well as pixel volumes
- OMF files
  - The output files from OOMMF micromagnetic simulations
  - Only the regular grid type can be loaded by Sasview
  - Only stores the magnetisation vector
- PDB files
  - A file type developed by the protein data bank
  - Stores locations of individual atoms
  - Nuclear SLDs can be extracted using the known scattering length and atomic volume of each element

In order to be of use the calculator needs to know the scattering length densities of the sample. The easiest way to specify these is as a regular rectangular grid of pixels, each with a constant nuclear and magnetic scattering length density.

This is the data type of the default data in the calculator (primarily useful for testing purposes) and can also be loaded from three different file types.

The sld file format is custom for Sasview and allows the real space position of any pixel to be stored, as well as the nuclear and magnetic scattering length densities, and the volume of the pixel. While not strictly enforced by Sasview – all sld files should be a regular rectangular grid of pixels. Irregular grids can be handled by vtk files as discussed later.

The OMF format allows only a magnetisation vector to be stored. While the OMF format allows for irregular grids to be stored, Sasview can only read in regular OMF grids.

The PDB file format specifies the position of atoms in space, and each of these is approximated by a pixel of the appropriate volume. Only nuclear scattering length densities are stored

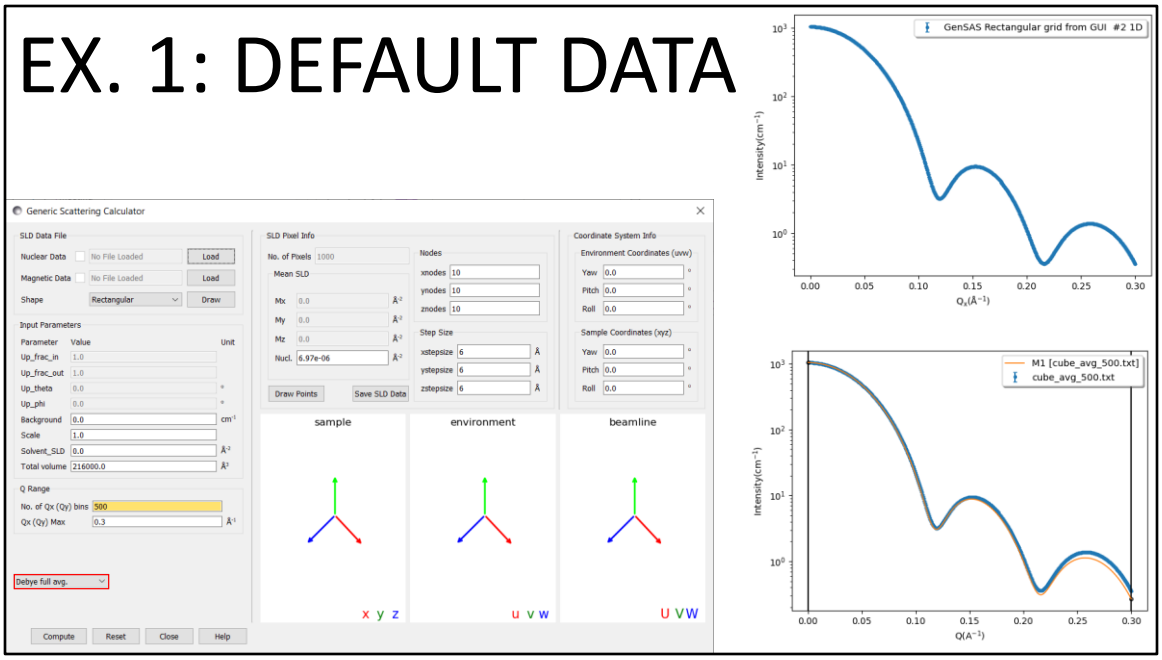
Any combination of these files can be loaded as the nuclear and magnetic scattering length density data – provided that they agree on the real space positions of the pixels

EX. 1: DEFAULT DATA

As a simple example of the calculator we can use the default data – which describes a 60x60x60 angstrom cube as 10x10x10 pixels. This default is useful for testing the functionality of the calculator and confirming that it conforms to analytical results.

Pressing calculate produces the expected 2D pattern, which can be compared to the analytical result produced by the fitting calculator.

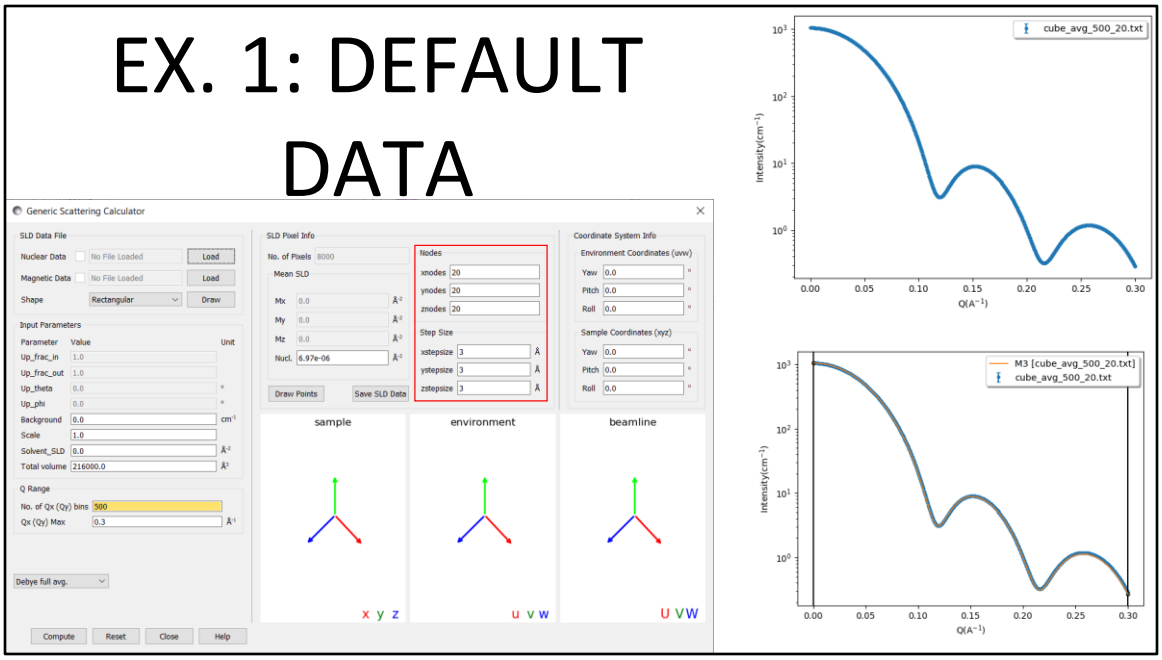
EX. 1: DEFAULT DATA



For grid type, nuclear only data such as this we can also perform an orientational average using the Debye formula. This produces the following 1D curve, which can again be compared to the analytical result.

The low discretisation of the sample, only 10x10x10 pixels, leads to a deviation between the analytical and simulated results.

EX. 1: DEFAULT DATA



By increasing the resolution we see a much closer fit, for example with 20x20x20 pixels the following pattern is seen.

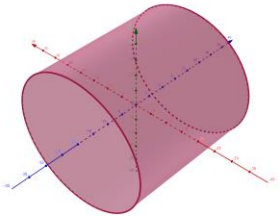
EX. 2: MAGNETIC CYLINDER

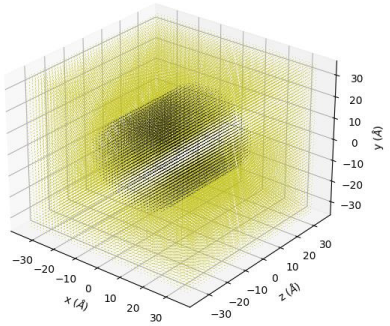
mag_cylinder.sld

X	Y	Z	N	Mx	My	Mz
-35	-35	-35	0	0	0	0
-35	-35	-33.6	0	0	0	0
-35	-35	-32.1	0	0	0	0
…						
9.29	13.6	-19.3	1e-6	0	0	1e-6
9.29	13.6	-17.9	1e-6	0	0	1e-6
9.29	13.6	-16.4	1e-6	0	0	1e-6

…

N = $1x10^{-6}$ Å^{-2}
$b_H M_x$ = 0 Å^{-2}
$b_H M_y$ = 0 Å^{-2}
$b_H M_z$ = $1x10^{-6}$ Å^{-2}





For any more complex situations we need to use a file to store the scattering length density data. For example we could create an sld file which describes a cylinder as shown (note the default unit of angstroms).

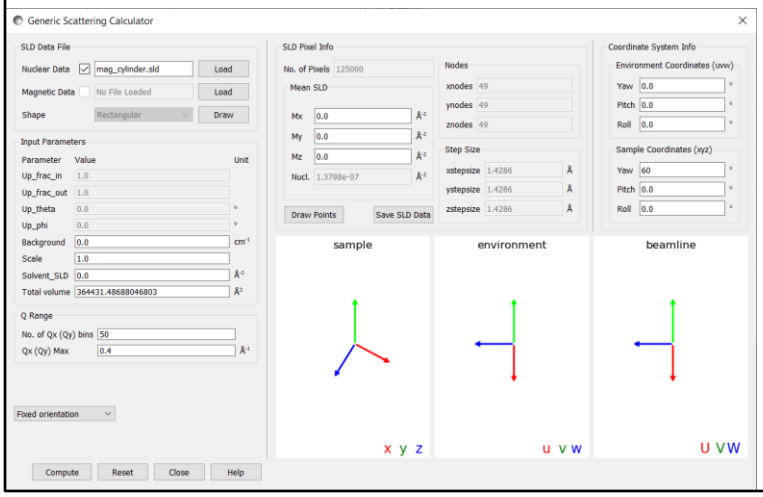
The cylinder has a radius of 20 angstroms and a length of 40 angstroms. It has a constant nuclear scattering length density of 1e-6 per square angstroms and a constant magnetic scattering length density of 1e-6 per square angstroms in the positive z direction (along the axis of the cylinder).

This sld file lists the positions of each pixel and the scattering length density, but leaves the volume of each pixel to be worked out, since it is constant across the grid. The allowed sld file formats are found in the updated documentation (currently on its own branch).

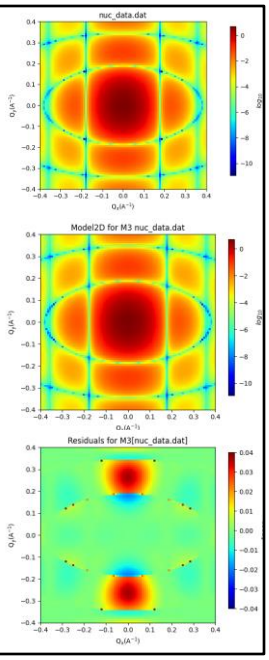
This sld file consists of a 50x50x50 pixel simulation box ranging from -35 to +35 angstroms in each direction

The large number of of 0 scattering length density pixels (yellow in the visualisation) does not slow down the program because they are stripped out prior to calculation.

EX. 2: MAGNETIC cylinder CALCULATOR AND RESULTS



$\chi^2 = 7.72 \times 10^{-5}$



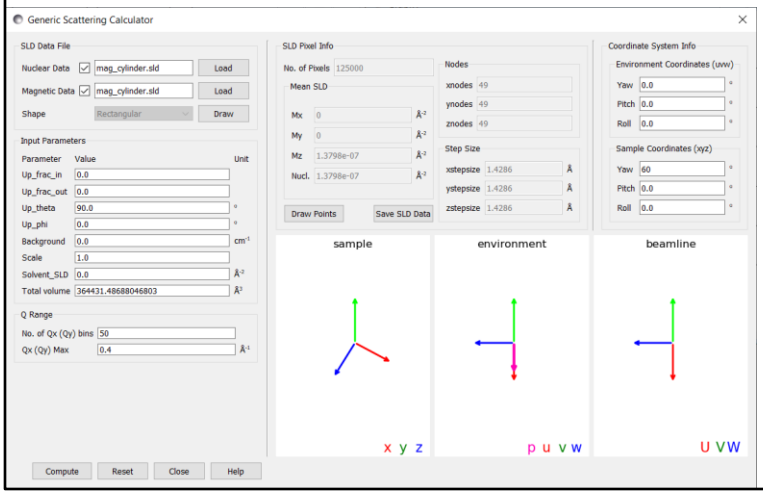
We first of all consider an unmagnetized cylinder by loading the file into the nuclear data tab only. Note that the magnetic scattering length density textboxes are still editable to allow a constant value to be specified.

While it was easiest to generate this file aligned with the axes – we may wish to know the scattering pattern at a different angle, which we can easily alter in the sample coordinates panel. Here, for example, the axis of the cylinder is at 60 degrees to the W axis

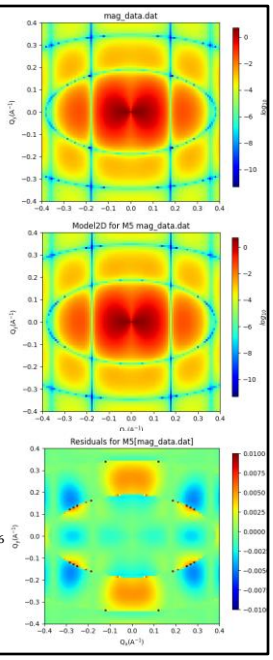
We can calculate the expected scatting pattern as before, as well as the analytical result from the fitting calculator.

We can also use the fitting tool to plot the residuals, and find the chi squared value of the fit, which comes out very low – as we would hope with such a small discretisation.

EX. 2: MAGNETIC cylinder CALCULATOR AND RESULTS

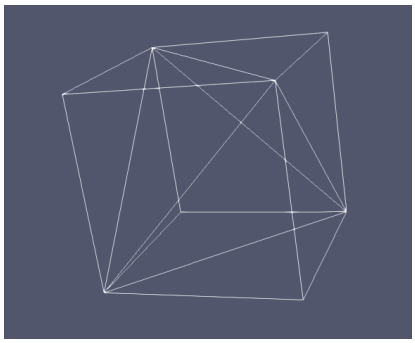


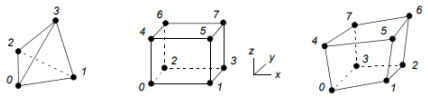
$\chi^2 = 4.98 \times 10^{-6}$



By loading in the magnetic portion of the file we can also see the results for the magnetised cylinder. Choosing the polarisation to be along the u axis we see the following result – which can again be compared to the analytical model, and the residuals plotted.

DATA TYPES II – element data





https://vtk.org/wp-content/uploads/2015/04/file-formats.pdf

File Types

- Legacy VTK files
  - Describe a broad array of different grid types
  - Sasview currently only supports unstructured grids where every element has same number of faces, every face same number of elements
  - Some combination of shown elements

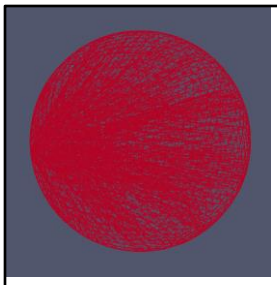
Advantages

- Significantly fewer discretisation artifacts in final output
- Can change the level of detail in different regions
- Allows finite element simulation output to be used by the calculator as input

At other times it may be easier to represent the scattering length densities on an irregular grid, such as output from a finite element simulation. This can allow areas of little interest to be described with much less detail, saving computation time.

Additionally for low resolution samples, this avoids discretisation errors, although at the cost of a more time-intensive algorithm.

Sasview supports these grid types in unstructured grid datasets of the .vtk files, the legacy file format of the visualisation toolkit, which is supported by many other programs, such as netgen, which was used to generate the files used here.



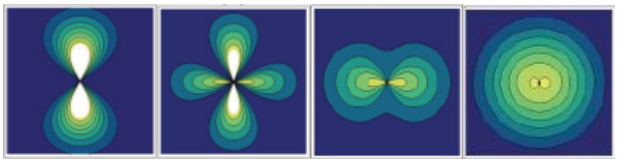
EX. 3: MAGNETIC SPHERE DATA TYPES

Polarisation along x axis, sample aligned with beamline – U axis ⇔ x axis

N(**r**) = N
b_H**M(r)** = (0, 0, M_x)

$$\frac{d\Sigma^{++}}{d\Omega}(\mathbf{q}) = \frac{8\pi^3}{V}\left(\left|\widetilde{N}\right|^2 + b_H^2\left|\widetilde{M}_x\right|^2 \sin^4\theta - b_H\left(\widetilde{N}\widetilde{M}_x^* + \widetilde{N}^*\widetilde{M}_x\right)\sin^2\theta\right)$$

$$R = \frac{\left|\widetilde{N}\right|^2}{b_H^2\left|\widetilde{M}_x\right|^2}$$

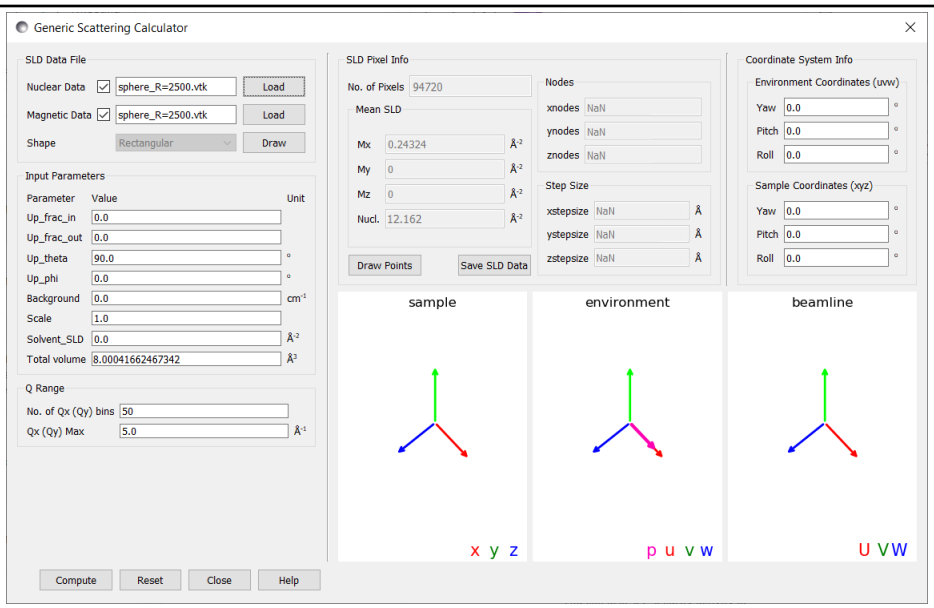


R: 0.0025 0.2025 4.0 2500

1. Analytical results derived in:
Observation of cross-shaped anisotropy in spin-resolved small-angle neutron scattering
Andreas Michels, Dirk Honecker, Frank Döbrich, Charles D. Dewhurst, Kiyonori Suzuki, André Heinemann
DOI: 10.1103/PhysRevB.85.184417

A simple example, and hence one with an analytical result, is a uniform sphere, with a constant magnetisation perpendicular to the beamline. The ratio of the nuclear to magnetic scattering length density (R) influences the shape of the scattering pattern – particularly in the ++ cross section, in which a cross shaped anisotropy appears for R values less than unity.

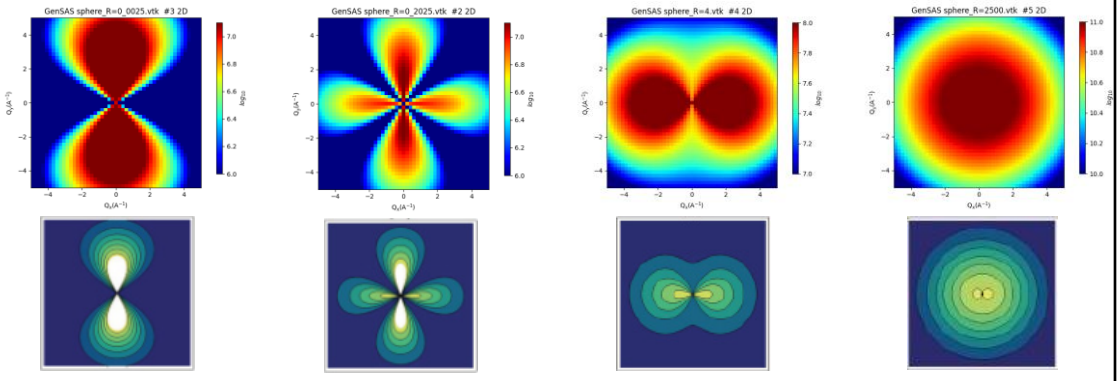
We will perform both a quantitative and qualitative comparison of the results



*composite image as expected for release version

As before we load in both the nuclear and magnetic components of the file and then compute the scattering pattern

RESULTS



max |err|: 0.74%
mean(|err|): 0.17 %

max |err|: 1.00%
mean(|err|): 0.17 %

max |err|: 0.74%
mean(|err|): 0.17 %

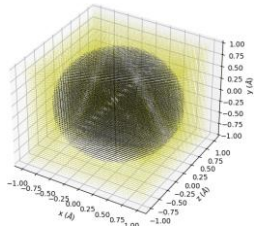
max |err|: 0.74%
mean(|err|): 0.17 %

Qualitatively we see a very close match between the analytical results and the plots produced by Sasview.

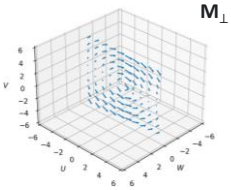
To do an analytical comparison we take the percentage error of each pixel and find the maximum and average value for each plot.

As can be seen these errors are relatively low. The main contributions to the error come from the points where the structure factor goes to zero. At these points small errors can become highly significant to the final result – giving large relative errors.

Rocking curves

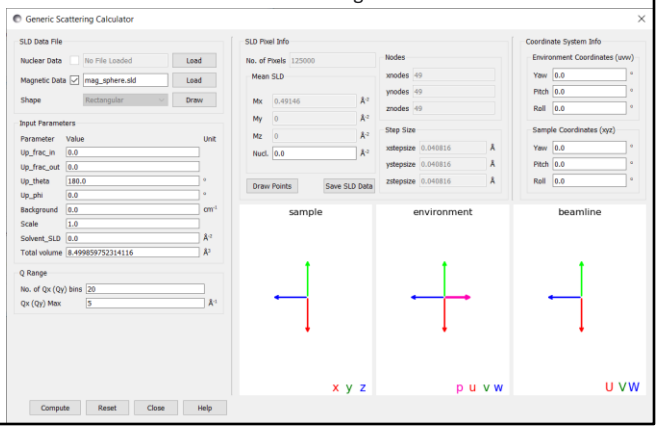


N = 0 Å^{-2}
b_H**M** = (1, 0, 0) Å^{-2}
P = (0,0,-1)
Rocking curve around the vertical axis



$\mathbf{M}_\perp$

$\mathbf{M}_\perp \cdot \mathbf{p} = 0$



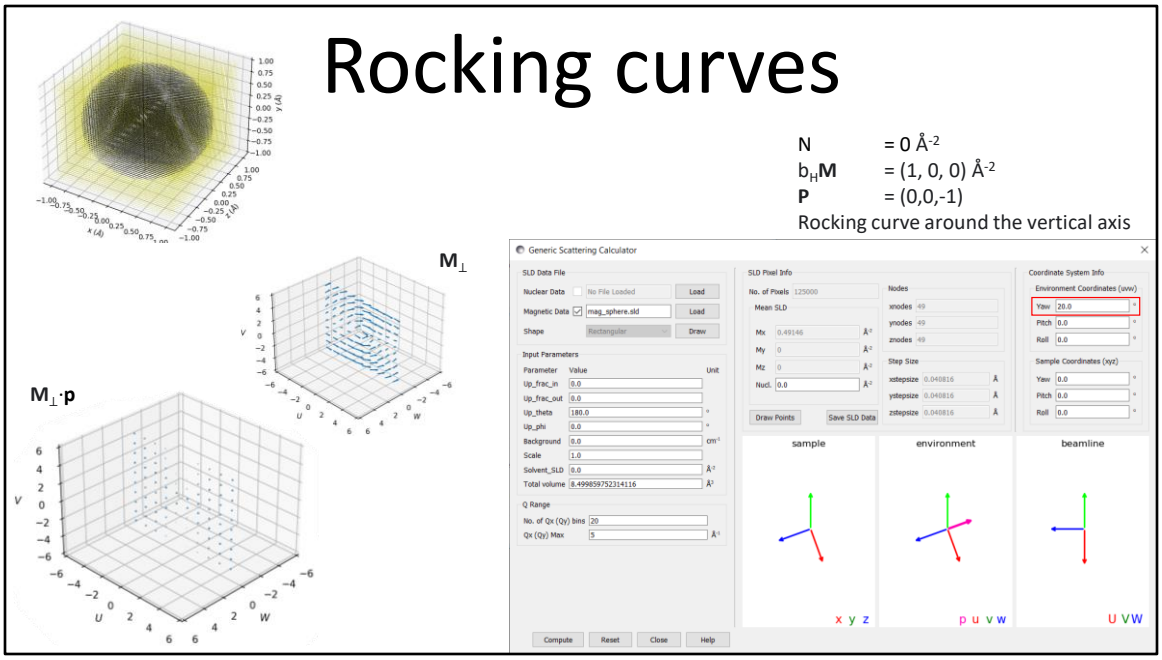
Another use of the coordinate interface is the ability to generate rocking curves. By creating small rotations about a central axis a series of diffraction images can be observed, from which a rocking curve can be generated.

Let's consider again the uniformly magnetised sphere – with a constant magnetic scattering length density in the U direction in beamline coordinates. We'll also set the nuclear scattering length density to 0 (relative to the solvent).

The polarisation will be along the W axis towards the target. The important part of the magnetisation is the perpendicular component to the Q vector – which we can plot as a vector field. This is non-zero but lies in the UV plane, so if we examine the ++ cross section where only the component of this parallel to the polarisation matters, we get no scattering intensity.

Rocking curves

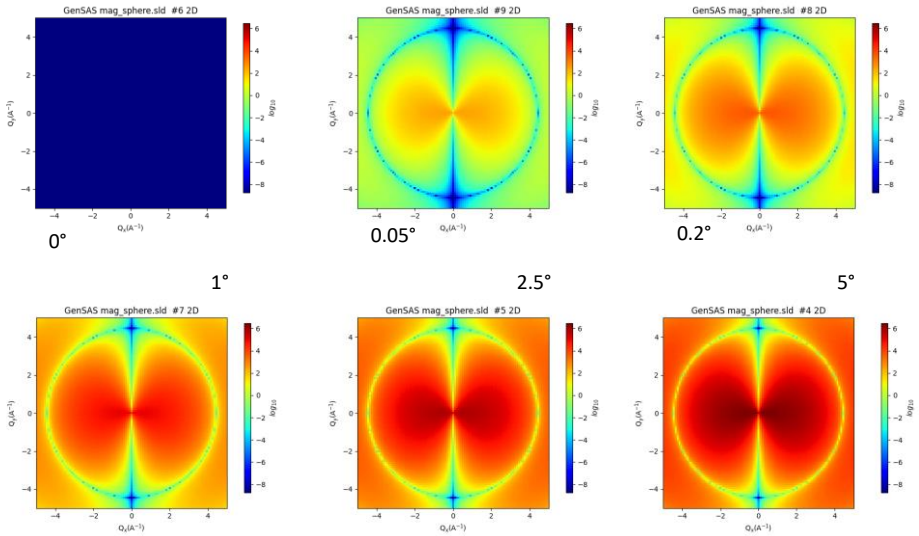
N = 0 Å^{-2}
b_H**M** = (1, 0, 0) Å^{-2}
P = (0,0,-1)
Rocking curve around the vertical axis



Now lets rotate the environment about the V axis (a yaw in the environment coordinates). Now the M perpendicular vector has shifted, and importantly there is now a non-zero component in the polarisation direction.

As a result as the yaw angle increases we should see a magnetic scattering pattern begin to appear

Rocking curve



0° 0.05° 0.2°

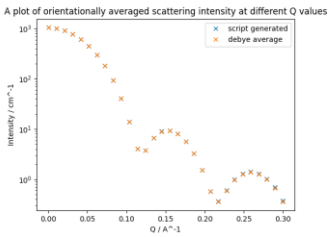
1° 2.5° 5°

The following images show exactly that behaviour. From these a rocking curve for a particular point could be calculated .

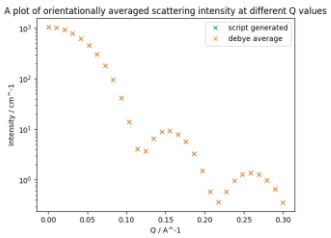
```python
import numpy as np
import math
from scipy.spatial.transform import Rotation
from sas.sascalc.calculator import sas_gen

STEPSIZE=6
NODES=1
ANGLESTEP = 10 # total samples = 4 * anglestep^3

def create_rotation_grid(theta_step, phi_step, chi_step):
    theta = np.linspace(0, 180, theta_step+1)
    phi = np.linspace(0, 360, phi_step+1)
    chi = np.linspace(0, 360, chi_step+1)
    THETA, PHI, CHI = np.meshgrid(theta, phi, chi, indexing="ij") # first index on theta, second on phi
    centre_theta = np.zeros((THETA.shape[0]-1, THETA.shape[1]-1, THETA.shape[2]-1))
    centre_phi = np.zeros((PHI.shape[0]-1, PHI.shape[1]-1, PHI.shape[2]-1))
    centre_chi = np.zeros((CHI.shape[0]-1, CHI.shape[1]-1, CHI.shape[2]-1))
    solid_angle = np.zeros((THETA.shape[0]-1, THETA.shape[1]-1, THETA.shape[2]-1))
    for i in range(THETA.shape[0]-1):
        for j in range(THETA.shape[1]-1):
            for k in range(THETA.shape[2]-1):
                centre_theta[i,j,k] = (THETA[i, j, k] + THETA[i+1,j, k])/2.0
                centre_phi[i,j,k] = (PHI[i, j, k] + PHI[i,j+1, k])/2.0
                centre_chi[i,j,k] = (CHI[i, j, k] + CHI[i,j, k+1])/2.0
                # solid angle = sin(theta)d(theta)d(phi), chi is evenly distributed so don't include
                solid_angle[i,j, k] = math.sin(np.radians(centre_theta[i,j, k]))*np.radians((THETA[i+1, j, k] -
 THETA[i,j, k]))*np.radians((PHI[i, j+1, k] - PHI[i,j, k]))
    return np.column_stack((centre_theta.flatten(), centre_phi.flatten(), centre_chi.flatten(), solid_angle.flatten()/(n
p.sum(solid_angle))))

Qs = np.linspace(3e-4, 0.3, 30)
points = np.linspace(0, STEPSIZE*NODES, NODES, endpoint=False)
pos_x, pos_y, pos_z = np.meshgrid(points, points, points)
pos_x = pos_x.flatten()
pos_y = pos_y.flatten()
pos_z = pos_z.flatten()
data = sas_gen.MagSLD(pos_x, pos_y, pos_z, np.full_like(pos_x, 6.97e-06))
model = sas_gen.GenSAS()
model.set_sld_data(data)
angles = create_rotation_grid(ANGLESTEP, 2*ANGLESTEP, 2*ANGLESTEP)
output = np.zeros_like(Qs)
next_frac = 0.01 # for printing progress - the next fraction of completeness at which to print
for i in range(len(angles)):
    r = Rotation.from_euler("ZYZ", [np.radians(angles[i, 1]), np.radians(angles[i, 0]), np.radians(angles[i, 2])])
    model.set_rotations(xyz_to_UVW=r)
    output += model.runXY([Qs, np.zeros_like(Qs)]) * angles[i, 3] # weights already normalised  no need to divide by sum
    if i/len(angles) >= next_frac:
        print(f"{next_frac*100: .1f}%")
        next_frac += 0.01
```

SCRIPTING

4000 samples



500,000 samples



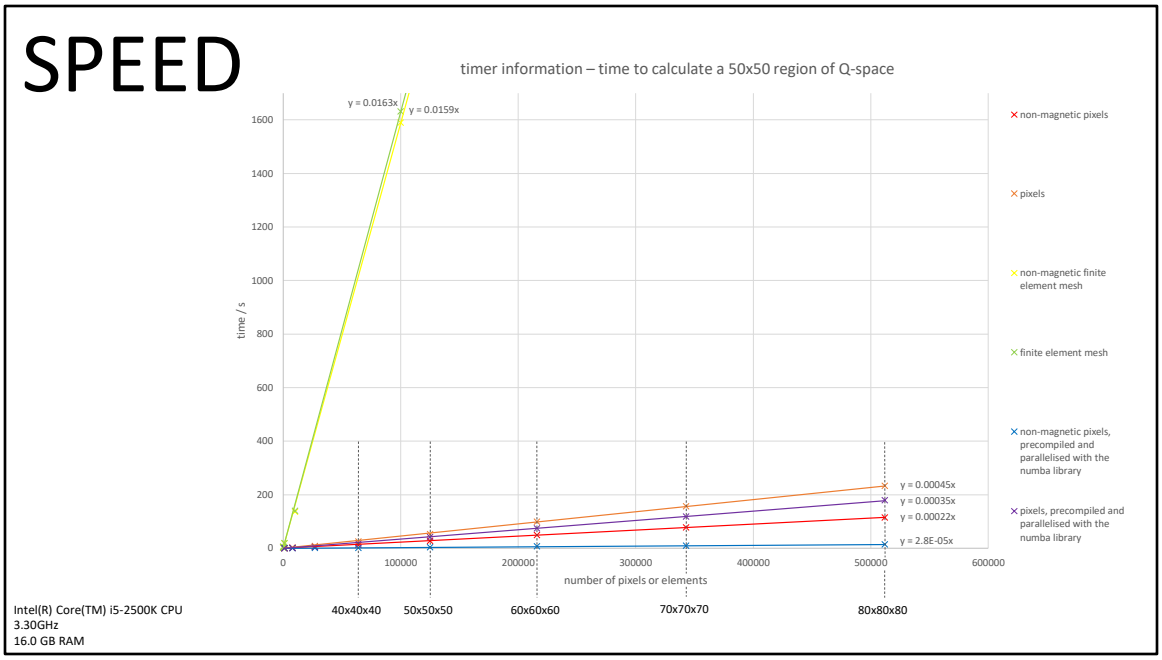
For more complicated tasks the GUI may not have the required functionality. In this case a python script can be used to interface with the backend of the calculator. A brief overview of this interface is provided in the updated documentation.

One example use is the calculation of an orientational average for a structure. The Debye average is only available for non-magnetic, grid type data. In this case the script creates a discretisation of 3D rotation space, and averages the desired model. Here we are using the default data of the calculator and comparing the results to the Debye average functionality. As can be seen for reasonable numbers of samples the results become very close, with the divergences occurring at higher Q values.

As before both of these methods are limited by the resolution of the model. The true value is substantially different from both, and we could improve our simulation as before by using a finer resolution grid.

This script could easily be altered to use a magnetic model or element type data – in which case it would not be possible to use a Debye average, and scripting the calculator is the only way to calculate an orientational average.

SPEED



Intel(R) Core(TM) i5-2500K CPU
3.30GHz
16.0 GB RAM

While the speed of the calculator depends on the computer on which it is being run, we can gain a rough estimate and a good idea of the relative speed. These tests were run on a desktop PC. The calculations were carried out over a 50x50 grid of Q space.

The grid type data runs relatively fast – with non-magnetic data running approximately twice as fast as magnetic data. Installing the python package Numba – which precompiles python code and parallelises certain operations dramatically improves the nuclear only code, and provide a reasonable improvement to nuclear and magnetic pixel data.

Element type data runs significantly slower for the same number of elements as pixels. If the data can be processed as grid type data without significantly increasing the number of pixels, it is worthwhile. Element type data is best used when large regions of the space have very little variation, and so the number of elements is low, compared to the number of pixels that would be needed to cover the same space without significant discretisation errors.

A potential future upgrade is to re-write the element-wise Fourier transform to be Numba compatible.

New Features

- Improved graphical interface
  - Ability to load separate nuclear and magnetic files
  - The default data can be computed for testing purposes
- Finite element Fourier transform
  - Allows irregular meshes to be used
  - Varying levels of resolution in the simulation box
  - Use output of finite element simulations
- Coordinate system
  - Rotate sample independently – do not need a new file for each rotation
  - Calculate rocking curves using environment rotation
  - Useful for scripting – for example orientational averages
- Small improvements
  - Addition of cancel button for long calculations
  - Fix to allow data to be saved as an sld file
  - Fix to allow output to be plotted as a theory curve
  - Addition of unit tests for the calculator

Future Features

- The improvement of the 3D graphical environment – both the coordinate systems visualisation, and the display of data sets using the draw functionality
- The implementation of a Numba-compatible finite element Fourier transform – to enable the parallelisation and precompilation of the algorithm
- Expansion of the types of irregular grid that Sasview can handle
  - Polygonal data can contain more complicated element types.
  - Handle multiple types of element in one file, e.g. tetrahedra and cubes