Scorpy Progress Report

Patrick Adams

1 Introduction

Scattering Correlation in Python (SCORPY) is a python package developed by members of the X-ray diffraction group in the department of Physics at RMIT University. The goal of the package is to supply easy to use tools for calculating and analysing scattering correlation data, with an emphasis for use with X-ray diffraction from protein crystals.

2 Object Overview

The following section outlines a set of python classes used in scorpy to conduct correlation analysis.

2.1 Vols

In scattering correlation analysis, 3D data sets frequently occur in the form of PADFs ($C(r_1, r_2, \Psi)$), Correlation functions ($C(q_1, q_2, \Psi)$) and Harmonic Correlation functions ($B_l(q_1, q_2)$). Vol objects are a representation of any 3D data set or space. In scorpy, all Vol-like objects inherit from the base Vol class, which contains properties that are shared between all Vol objects. Children classes that inherit from the Vol class, such as the CorrelationVol and BlqqVol, contain additional methods and properties unique to that type of data representation. Classes for all vol objects can be found in the scorpy/vols directory. Each file contains a single class for each type of vol used by scorpy, with an extra file volspropertymixins.py. This files contains property decorated methods for each vol that are "mixed in" with the associated class. For example, the methods contained within VolProps in volspropertymixins.py could be kept in the actual Vol class in vol.py, but are included in the separate file to reduce clutter.

2.1.1 Vol

Code for the Vol class can be found in scorpy/vols/vol.py. Vols are defined by a series of parameters describing the shape and domain of the data array that the Vol handles. Parameters nx, ny, nz define the number of bins for each axis of the data array. Parameters xmax, ymax, zmax define the maximum value that data in each axis can hold. Parameters xmin, ymin, zmin define the minimum value that data in each axis can hold. Parameters xwrap, ywrap, zwrap indicate if the axis is periodic (True) or not (False, default). Wrapping

axes are useful for angular coordinates. The parameter comp defines whether the array stores complex numbers (True) or not (False, default). With these parameters, we initialise a numpy array with shape (nx, ny, nz) of zeros.

Using the @property decorator, all parameters should be pseudo-write protected. There are also extra properties provided by the VolProps class. xpts, ypts, and zpts provides the value of the sampled point at the centre of each bin, for each axis. This will change depending on the wrapping of each axis.

For example, if vol.xwrap=False, vol.xmin=0, vol.xmax=1, and vol.nx=10 then vol.xpts=[0.05, 0.15, 0.25, ..., 0.95]. The left wall of the 0th bin is at vol.xmin, and the right wall of the last bin is at vol.xmax. Conversely, if vol.ywrap=True, vol.ymin=0, vol.ymax=180, and vol.ny=6 then vol.ypts=[0, 30, 60, ..., 150]. The left wall of the 0th bin is at -15, and the right wall of the last bin is 165. Since the axis is wrapped, the left wall and right wall are the same value (180-15=165). This is used for consistency with the other python packages (pyshtools).

We also have properties describing the bin resolution, width, or distance between sample points. These are accessed with vol.dx, vol.dy and vol.dz. The actual data can be accessed with the vol.vol. The data is not write protected, but any replacement asserts that the replacement array shape is the same as the original array shape.

Vol objects also contain various methods that are useful across all datasets. The get_eig method calculates the eigenvectors and eigenvalues of the xy axe slices of the data. The extra parameter herm flags the method to calculate the hermitian matrix calculation (True, default) or not (False). The hermitian calculation is faster, but requires real valued vols. Note that for CorrelationVol and BlqqVols, $V(q_1, q_2, \Psi) = V(q_2, q_1, \Psi)$, so the hermitian condition is satisfied.

The convolve method will apply a Gaussian convolution over the data, which is useful for blurring peaks. The kernel is described by a domain kern_L, resolution (number of bins) kern_n, and standard deviations in x, y and z (std_x, std_y, std_z). These parameters control the size and extent of the blurring. Note that edge effects are not well defined and windowing can occur at the edges of the data volume. It is advised that this method only be used to ease visualisation, and and has not been characterise for use as part of a processing pipeline.

For CorrelationVols, BlqqVols and PADFVols, it can be useful to extract a diagonal slice through the volume through the $q_1 = q_2$ or $r_1 = r_2$ plane. More formally, this would be extracting the $x = y$ plane of the data set. The method get_xy asserts that the number of bins in x an y are the same, and that the axis domains are the same (xmax=ymax, xmin=ymin), and then extracts the x=y plane from the data volume.

There are various plotting methods included in the class, for example, plotting the xy plane (plot_xy), plotting the sum through an axis (plot_sumax), plotting a particular slice through a given axis (plot_slice), or producing a line plot through an axis at a given point (plot_line).

The method ls_pts creates a list of points for every intensity value in the data array. This returns an nx4 array of values, where the first 3 columns are the sampled positions of the point from xpts, ypts, and zpts, and the last column of the array is the intensity of the volume at that position.

Finally, the most useful methods of the volume objects are the save and load methods. When a volume object is created, and it's data manipulated, the volume can be saved in a .dbin format. The saving process is as follows;

The array is flattened with the numpy array method flatten(), and saved with the numpy array method tofile(). Then, a log file is created that contains the relevant properties of the of the vol object such as number of bins, axes domain, wrapping, etc. Since the log file is used to recreate the array from the dbin, it is critical that the log file be kept in the same directory as the dbin. The filenames are also important. If the dbin is saved as [filename_tag].dbin, then the associated log file will be called [filename_tag]_log.txt. If either filename_tag is changed, the change has to be reflected in the other filename. There is an internal _save_extra method that is used by children of the vol classes, such as CorrelationVol and BlqqVols, which save extra information specific to that class.

To load a vol object, exclude all input parameters described above and only use the path parameter. This is a string describing the location of the dbin to be loaded. If the log file is in the same directory, the parameters are parsed and the vol object in initialised. The dbin file is loaded with the numpy function fromfile(), and reshaped from the flat array the with the extracted nx, ny, and nz values from the log.

2.1.2 CorrelationVol

2.1.3 BlqqVol

2.1.4 SphericalVol

2.2 Readers

2.2.1 CifData

2.2.2 PeakData

2.2.3 ExpGeom