**Design considerations for Horace type interface**

We would like to create algorithms that mimic the functionality and syntax of Horace commands

1. gen\_sqw function

*gen\_sqw (spe\_file, par\_file, sqw\_file, efix, emode, alatt, angdeg, u, v, psi, omega, dpsi, gl, gs)*

- based on ConvertToMD

- obviously we will get rid of spe files and par files. Instead we should use processed nexus files. This is very much suited to autoreduced data at SNS

- for processed nexus files, there is no reason to include efix

- the workspaces must be in units of energy transfer

- alatt, angdeg, u, v are used to set the UB matrix. One can set the UB matrix before. If not empty, it will override and set the UB again

- omega, dpsi, gl, gs are angles for goniometer settings. This should be in principle set on the input workspaces beforehand. If omega is set, it should override the individual goniometer settings for each individual workspace. Dpsi, gl, and gs are additional goniometer rotations. **TODO:** decide if we want to use any of the angles. Also, if omega is not set, do we use dpsi, gl, and gs?

- there are some optional parameters grid\_size\_in that maps into the “SplitInto” parameter in ConvertToMD, and urange\_in, which maps into “MinValues” and “MaxValues”

Proposed workflow (1 algorithm):

 - load each individual file. I propose that we require to be in units of energy transfer

 - set the UB matrix, if desired

 - Optionally set the goniometer

 - calculate limits (if not given)

 - ConvertToMD

 - SaveMD

 - at the end, use MergeMDFiles

Question:

 - do we want to enforce that all input workspaces have UB matrices/ goniometers set?

2. cut\_sqw function

*cut = cut\_sqw (data\_source, proj, p1\_bin, p2\_bin, p3\_bin, p4\_bin, '-nopix', filename)*

- data source must be an MDEvent workspaces

- this will be a wrapper around BinMD/SliceMD

- the proj object will be a slightly modified table workspace. The python bindings will allow Horace syntax like proj.u= “1,1,1”

- we can generate a proj object from an algorithm, something like proj=GenerateSQWProj(arguments). If no arguments given, we will use sensible defaults

- I propose we add proj.w, the third projection axis, since we can do non-orthogonal axes. If not given proj.w is calculated as the cross product of proj.u and proj.v

**TODO:** decide if we want to have the -nopix option on by default (will use BinMD), and use SliceMD for the “-withpix” option

Proposed workflow (2 algorithms):

 - first algorithm to generate projection

- second algorithm to produce the cut

3. plot

*plot(cut)*

- to be done at a later date

- it will be dependent on the dimensionality of the MD workspace

- plotMD is not exposed to python (yet). It should be similar to plotSpectrum. This would be the preferred way to plot 1D, since it allows control in python

- for more than one dimension, the only option that allows python control (for now) it's SliceViewer. Need to check if we can do plot2D (nicer printed graphics)

- for 3D or more we need to enable python control for vates

4. Other issues

- need to implement symmetrizing

- need to implement fitting