Excitations Data Analysis Training Course – Worksheet 3

Horace Introduction

Horace is a program to view and analyse single crystal inelastic neutron scattering data from time-of-flight (ToF) spectrometers. These machines have a large area detector and measure the neutron's ToF in order to determine their velocity (energy). Combined with a way of defining either the neutron's energy before or after it scatters from a sample the ToF information allow the neutron's energy transfer to or from the sample to be determined. The location of the detected neutron on the 2D area detector then allows two of the three components of the momentum transfer vector to be determined.

Horace is based on the observation that depending on the orientation of a single crystal sample with respects to the incident beam, the projection of the 2D detector surface on the 3D momentum transfer space changes. Thus a 4D $S(\boldsymbol{Q},\omega)$ dataset can be built up from multiple measurements of a single crystal sample at different orientations. Horace is designed to perform such a recombination and then to quickly access and rebin the data in 3D, 2D, or 1D for further analysis. The data structures which permits this quick access is describe in more detail in the Horace paper [1].

Horace Commandline

Yesterday we covered the GUI tools available with Horace – the Horace Planner and the Horace GUI. However, Horace is mainly a commandline (CLI) driven program, so to do more than the basics you will need to use the Matlab CLI. There is a hint-sheet / aide memoire if you're not familiar with Matlab.

Making a "fake" dataset

Yesterday we covered using the Horace Planner to determine the run parameters for an experiment. The Horace Planner is good for systems with orthogonal axes but is harder to use for lower symmetry crystals. Also, it may not fully show the full extent of the ($\boldsymbol{Q}$,ω) coverage you can get for a given **Ei** and **psi** range. For these details, you can generate a fake dataset which mimics the dataset you would generate in your experiment. You can take cuts and slices from it just as you could the real data, so you can explore not just whether a particular reciprocal lattice point is covered, but also what is going on along off-symmetry directions nearby. The intensity scale is given by the average value of psi that contributed to a given bin.

The disadvantage of this approach is that if you wish to change the values of Ei and/or range of psi for your fake dataset, you have to spend a few minutes regenerating it. The advantage is that you get much more information about what you will measure in your experiment. Also, as described above, these routines are essential for planning measurement of systems where the crystallographic axes are non-orthogonal.

The following script will generate a fake dataset similar to the one used in the Horace Planner exercise yesterday:

```
% Name and folder for output "fake" generated file
sqw_file = 'my_fake_file.sqw';

% Instrument parameter file (may be in another location to this)
par_file = '4to1_102.par'];

% u and v vectors to define the crystal orientation
% u||ki, uv plane is horizontal but v does not need to be perp to u
u = [1, 0, 0];
v = [0, 1, 0];

% Range of rotation (psi) angles to cover in simulated dataset.
% (psi=0 when u||ki)
psi = [0:5:90];

% Incident energy in meV
efix = 401;
emode = 1;   % This is for direct geometry (set to 2 for indirect)

% Range of energy transfer (in meV) for the dataset to cover
en = [0:5:360];

% Sample lattice parameters (in Angstrom) and angles (in degrees)
alatt = [2.87, 2.87, 2.87];
angdeg = [90, 90, 90];

% Sample misalignment angles ("gonios"). [More details tomorrow].
omega=0; dpsi=0; gl=0; gs=0;

% This runs the command to generate the "fake" dataset.
fake_sqw (en, par_file, sqw_file, efix, emode, alatt, angdeg,...
                     u, v, psi, omega, dpsi, gl, gs);
```

You can copy and paste the above code into a Matlab script window (if one is not open, right-click on the left panel and select New → Script and then double click on the file created; or type `edit myscript.m` in the Matlab command window) and the press `Ctrl+Enter` to run it.

Once the fake dataset has been created, you can use the `cut_sqw` command to make cuts from it. This is exactly the same as with the Horace GUI yesterday. An example script to make the same 3D cut as the first example yesterday is:

```
% First we define the projection axes (the u, v and w vectors)
% Type is Q units for each axis and can be either 'r' for r.l.u. or
% 'a' for absolute (A^-1). E.g. 'rar' means u and w are
% normalissed to in r.l.u, v in A^-1.
proj = projaxes([-1,-1,1], [0,1,1], 'type', 'rrr');

% Now make a cut of the fake dataset.
% The four vectors indicate either the range and step (3-vector) or
% the integration range (2-vector), with units defined proj.type
% The following makes a 3D cut with axes u, v and energy
% (first, second and fourth vectors are 3-vectors),
% integrating over w between -0.1 and 0.1.
% '-nopix' indicates to discard the pixel information and create
% a dnd (d3d) object.
my_cut = cut_sqw(sqw_file, proj, [-3,0.05,3], [-3,0.05,3], ...
                 [-0.1,0.1], [0,4,360], '-nopix');

% Now plot the 2D cut.
plot(my_cut);
```

Make a 2D Q-energy slice with a Q-axis of (h,0,0) around the (2,0,0) position, and repeat along other equivalent positions such as (0,2,0), (0,0,2), etc.

Which of the equivalent directions would be best to use here?

Tip: to keep multiple plots on the screen, use the ‘keep’ menu item on a plot.

Now try to make a 2D Q-energy slice with a Q-axis of (h,h,h) around the (2,0,0) position.

Hint: Use a different projection, and the **u** and **v** vectors do not have to be orthogonal.

Generating SQW files

Yesterday, we saw how to generate a 4D S(***Q***,ω) dataset using the GUI – now we will look at how to do it using the CLI. The command is `gen_sqw` and it has a very similar syntax to the `fake_sqw` example above.

Use help gen_sqw and the above example to write a script to create the iron.sqw file from the nxspe files in the `/home/dl11170/edatc/data` folder.

Hint: You need to generate a cell array of file names. You can use a `cellfun` and an anonymous function to do it using:

```
runs = 15052:15097;
data_path = '/home/dl11170/edatc/data';
spe_file = cellfun(@(c) fullfile(data_path, ...
      ['map' num2str(c) '_ei400.nxspe']), num2cell(runs), ...
      'UniformOutput', false)
```

Alternatively, you could also use a loop to do the same thing.

The other parameters are the same as we used yesterday in the GUI:

u – 1 0 0
v – 0 1 0
psi – 0 to 90 in 2 degree steps
lattice pars – 2.87 2.87 2.87
lattice angles – 90 90 90
Ei – 401
omega, **dpsi**, **gl**, **gs** – all zero

To get help, type `help gen_sqw` or `doc gen_sqw` in Matlab. This will provide details of the syntax of the command that we will use to combine the files. Alternatively look on the Horace website: http://horace.isis.rl.ac.uk/Getting_started#Creating_an_SQW_file

Create variables that will be used as inputs for this function – e.g. ei=401. (note the variable `emode=1` should be used – this specifies that the instrument (MAPS in this case) is a direct geometry machine). Run the command, and you should get an sqw file within a couple of minutes.

Note for experts

During an experiment the number of data files will increase as time goes by. To avoid having to regenerate the entire sqw file each time you wish to look at the latest data, you can use the command `accumulate_sqw`. The inputs to this are essentially the same as for gen_sqw, but instead of a list of measured psi and spe files, you provide a list of psi that you plan to measure and a list of spe files that will eventually exist. The routine will then check which files do actually exist, and make an sqw file out of those. If you then run the routine again later only the new data will be processed, and then appended to the existing sqw file, saving time.

Making cuts and slices and then plotting them

Before we make a cut or slice, we need to define what will be our viewing axes. These do not have to be the same as (or even related to!) the **u** and **v** vectors that defined the scattering plane. We make this definition by creating a object called `proj` with properties `u, v, uoffset, type`, and `nonorthogonal`.

The first two define a set of viewing axes in Q (3-element vectors) – the 3rd viewing axis is the cross-product of `u` and `v`. `uoffset` is a 4-element vector that defines an offset for the viewing axes, and `type` is a string that defines whether you wish to use reciprocal lattice units or inverse angstroms for each of the axes. The option `nonorthogonal` is a special option, whose use will be explained in an extension exercise. You can ignore it for the moment as by default it will be assumed to be false. (`uoffset` is also assumed to be `[0 0 0 0]` so it could also be ignored).

Let us define a structure proj1 as follows:

```
u = [1, 0, 0];
v = [0, 1, 0];
proj1 = projaxes(u, v, 'uoffset', [0, 0, 0, 0], 'type' = 'rrr');
```

You can get information on the project axis class using `help projaxes` or `doc projaxes`.

1. Create another proj structure array, with zero offset and reciprocal lattice units, that has (1,1,0) and (-1,1,0) as the viewing axes.
2. Now make a volume cut using `cut_sqw` (see this web page for the command syntax or type `help cut_sqw`) using the `proj` you just created. The range of Q along (1,1,0) should be -3 to 3 in steps of 0.05 (i.e. [-3,0.05,3]). Along (-1,1,0) the Q range should be the same, and along L we wish to integrate between -0.1 and 0.1. The energy axis should cover the range 0 to 360 meV in 4 meV steps. Select the option `'-nopix'` for now (this discards the pixel information and reduces memory but means you cannot re-cut from this cut).
3. Once you have created the volume, plot it using the plot command. This will pull up the sliceomatic window, which you can use to view orthogonal slices through the data: click on one of the slider bars to get a plane, and then drag the arrow up and down the slider bar.
4. Make a 2d slice from the data file using the same `proj` that has the same range and step sizes along (1,1,0) and energy. Integrate L between -0.1 and 0.1 again, and integrate along (-1,1,0) between -1.1 and -0.9. This time do not use the `'-nopix'` option. Plot the result.
5. Now make a 1d cut from the data file using the same proj. Keep the same range and step as before along (1,1,0), and integrate energy between 130 and 150 meV. Keep the other integrations the same as for your 2d slice. Plot the result once again.
6. Re-run the command above, this time timing it using the Matlab tic and toc commands i.e.

```
tic;
<Horace commands>;
toc
```

7. Now try taking a cut from a cut, i.e. rather than taking a cut from the file you can extract data if they already form part of another slice or cut. See this page for details of the syntax. In our case we want to take a cut from the slice you took in the exercise 4, keeping the binning the same along the first axis and integrating from 130 to 150 along the second axis.

8. Repeat the above using the `tic; <command>; toc` syntax as in exercise 6. Plot this cut and compare it to that which you generated in exercise 6. Are they the same? How long did it take to execute the command in exercise 6 compared to this exercise?

Basic customisation of plots

You will have seen in the plots you created in the previous section that the Q axes we chose went beyond the extent of the data in some cases. You can squeeze the axes to just enclose the data using the `compact` command: http://horace.isis.rl.ac.uk/Reshaping_etc#compact

Run this command on your 2d slice from exercise 4 above, and then plot the result.

Use the `smooth` command (http://horace.isis.rl.ac.uk/Reshaping_etc#smooth) to apply some smoothing to your 2d slice, and plot the result.

Answer: You could get an error if you made the cut you want to smooth without the `'-nopix'` option – this is because smoothing only affects the rebinned data, there is no consistent way to smooth the pixel data. Hence smoothing is only allowed on `dnd` type objects (without pixel information).

Try this with the default options, and then try using a Gaussian smoothing with varying widths, to see what this does.

Use the `lx`, `ly`, `lz` commands to modify the length of the axes of your plots – specifically put the x-limits as -1.5 to 1.5, the y-limits as 50 to 250 and the colour limits as 0 to 0.5 for your 2d slice plot.

(Note that you can reset the limits back to their original values by just typing lx, ly or lz with no additional arguments).

You will have noticed that every time you make a new plot of an object with the same dimensionality as that of a previous plot on-screen, the old plot is replaced by the new one.

Use the `keep_figure` command to retain a figure that will not be overwritten.

Try using the `xycursor` command on one of your plots.

What does it do?

Answer: Creates a cursor on the screen which lets you put text or coordinates on the plot.

A variant on this command is xyselect. Try it.

Plotting data with non-orthogonal axes

The option `proj.nonorthogonal` (set to be true or false; default false) is for optional use only when the lattice angles of the material are not all 90 degrees. If it is set to be false for such a system, the projection axes will be constructed to be orthogonal, with the first lying along u, the second in the plane of u and v, and the third perpendicular to this plane. If instead you wish to use the correct non orthogonal reciprocal lattice the option is set to be true. In this case the axes used will be those you define, however the images of plots will still be plotted as if these axes were orthogonal, so there will be a shear transformation of the image.

1. Take a cut from `/home/dl11170/edatc/data/upd3_elastic.sqw`, with constant energy in the range -2 to 2meV, and with L=-0.1 to 0.1. Note the proj.nonorthogonal option should be set to false by default. The remainder of the proj array should specify the (H,K)-plane for viewing data.

   Hint: Use an empty vector [] for the H and K axes to get the full range.
2. Make the same cut, with the nonorthogonal option set to be true for the (H,K)-plane. Note the differences between the two images – in particular the anisotropy of the peaks in both.
3. Take 1d cuts through the peaks along H and K in both projection bases. Compare and contrast the widths, and confirm the differences between the images make physical sense.

Correcting for sample misalignment

It is quite common to find after you have accumulated sufficient data that your sample was not perfectly aligned in the instrument. This can be seen by making slices or cuts at the elastic line in your Horace data. Horace provides a set of utilities to characterise and correct for such sample misalignment, and the following set of exercises provide a walk-through of how to use them.

1. Make a series of constant energy slices centred at 0 meV (+/-5 meV is a good integration range) to find out what Bragg peaks our example dataset covers. Make a list of 5 Bragg peaks (not all parallel to each other!) in a 5-by-3 matrix. Make sure you keep these slice plots for later.
2. Use the routine `bragg_positions` to make and then fit transverse and radial cuts through these peaks. Note that the input parameters to this routine are rather complex, so read the help quite carefully before starting. You should use a '`gauss`' lineshape, and the '`bin_absolute`' option in pretty much all cases.
3. You can examine the results of fitting the radial and transverse peaks using the routine `bragg_positions_view`. Before progressing to the next step it is imperative that all of the fits are good, otherwise your correction may end up making things less correct! Particular attention should be paid to the length and step size of the cuts to ensure reasonable statistics and a good fit.
4. Use the routine `refine_crystal` to obtain the alignment correction matrix (type help `refine_crystal` to get the syntax). In order to preserve the known cubic crystal structure, use the options `fix_angdeg` (to keep all lattice angles 90 degrees) and `fix_alatt_ratio` (to ensure the cubic lattice parameters a=b=c). This should give you a matrix called `rlu_corr`, that tells you how to modify the axes u and v (and their cross-product) to get the correct alignment.
5. Use the routine `change_crystal_horace` to apply this change to your sqw file (so you do not have to regenerate the whole dataset again).
6. Remake your constant energy slices, and compare what you have now with the plots you saved in part 1. Hopefully the Bragg peaks will be better aligned than they were before!
7. You can also use the results obtained above to work out the values for the goniometer offsets to be used for a correctly aligned sample in a regeneration of the sqw file using `gen_sqw`. The routine you need is called `crystal_pars_correct`

Extension question: Can you think of any circumstances in which it would actually be beneficial to deliberately misalign your sample and then apply the above post-hoc corrections? [Hint: Consider the symmetry of both the sample and the spectrometer]

Advanced Plotting

Dispersions (a.k.a. “Spaghetti”) plots

You will probably have seen band structure or phonon dispersion calculations that show the dispersion along several high symmetry directions. Horace allows you to make equivalent plots from your data using a routine called `spaghetti_plot`. Use the `sqw` file that you created for iron previously.

1. Make a basic dispersion plot that goes along the trajectory (1,-1,0) -> (2,0,0) -> (1,1,0) -> (1,-1,0).
2. Play with the qbin and ebin options to match the step sizes approximately to those used earlier.
3. Add labels (of your choice!) to the plot.
4. Save the slice along each trajectory into an sqw object, then plot these separately.

Useful Horace Plotting Commands

The commands in the table below can be used to adjust features of plots to make them more presentable.

`lx, ly, lz`	These are Horace-specific limit commands (similar to Matlab-native `xlim`, `ylim`, `zlim` and `caxis`) – because of customizations with Horace 3D, 2D and 1D plots, the Matlab standard commands don't work well and we recommend to use these instead. E.g. `lx(0, 2)` will set the x-limit to between 0 and 2.
`amark, acolor`	Horace-specific commands to change the marker type and color for the *next* 1D plot. These are similar to arguments to the Matlab-standard `plot` command, but again because of Horace-specific customizations of the 1D plots, it is recommended to use these instead.
`pp, pl`, etc	Please see: http://horace.isis.rl.ac.uk/Plotting#One_dimensional_plots for a list of Horace-specific 1D plot commands. These should be used in preference to the standard Matlab plot. E.g. to plot a data in square red marker and a fit line in blue use: `amark('s'); acolor('r'); plot(w1d); acolor('b'); pl(wfit);`
`title`	This sets the title of the current active plot, e.g. `title('My Plot')`
`xlabel, ylabel`	Sets the title of the x- or y-axis. E.g. `xlabel('Energy Transfer')` Additional parameters can be added to this, for example to change the font size, `xlabel('Qh', 'FontSize', 16)`. Type `doc xlabel` for more help. LaTeX can also be used in these and in the title commands – but you may have to set `xlabel(…, 'interpreter', 'latex')`.
`colorslider, colorbar`	colorslider is the Horace-specific widget which includes a standard Matlab colorbar with two editable text boxes to allow you to type the limits in – however, it does not look good in a plot output. `colorslider('delete'); colorbar` will replace the colorslider with a colorbar.
`text, annotation`	Matlab-standard commands which can add text or annotations to a plot. Type `doc text` or `doc annotation` to get help.
`print`	This saves a graph/figure to a file. E.g. `print('fig1.pdf', '-dpdf')` will save the current active figure to a pdf. Type `doc print` to get help.
`gcf, gca`	Matlab-standard commands to return the handle to the current figure (window, `gcf`) or axes (`gca`)

Use the above commands to make a nice 2D and 1D Horace plot.

References

[1] R.A. Ewings, A. Buts, M.D. Lee, J. van Duijn, I. Bustinduy, T.G. Perring, *Horace: Software for the analysis of data from single crystal spectroscopy experiments at time-of-flight neutron instruments*, Nucl. Instr. Methods Phys. Res. A, **834** (2016) 132.