

Reciprocal space mapping with fast_rsm

[Current issues with code](#) - table to keep track of bugs with the fast_rsm code

[Example datasets for testing and benchmarking fast_rsm mapper](#)

Intro

fast_rsm is a feature-rich piece of software that is designed to map raw detector images into reciprocal space. Scattering \mathbf{Q} -vectors are individually calculated per pixel per detector image; their corresponding intensities are individually corrected for polarisation, solid angle and transmission errors.

fast_rsm takes, as an input, detector data (images typically in the .tiff or .hdf5 formats) and metadata (containing e.g. motor positions during a scan, which is normally stored in .nxs and/or .dat files).

Note that while *fast_rsm* aims to only natively support direct mappings into 3D spaces, it is the aim of *fast_rsm* to support every physically relevant three-dimensional coordinate system.

Q: Why wouldn't you just map directly into Q_{xy} - Q_z / $Q_{||}$ Q / $I(Q)$ / $I(2)$ / $I(L)$ / insert favourite coordinate system here ?!

A: This would involve significantly expanding the scope of *fast_rsm* – allowing the core of the code to map to lower dimensions would introduce a significant headache with almost no reward. Instead, *fast_rsm* provides some convenience functions that can be used to downsample your 3D volumes into the above-mentioned lower dimensional coordinate systems.

Q: Does this introduce an error?

A: Yes. This error will be larger for smaller reciprocal space volumes. If this error is problematic, consider using the map_per_image option (discussed below) to compute exact reciprocal space maps.

Setting up SSH connection

Prior to loading and using the fast_rsm package, you will need to setup your SSH connection to the wilson server which is used to submit jobs to the computer cluster.

This is done with the following steps, note that commands to enter are the text after the \$ symbol

- log into a workstation with your fedID
- Create a fresh ssh key with the commands:
 - use command `$ ssh-keygen`
 - will ask to provide a name for the key pair, go with default option id_rsa `$ /home/<fed12345>/.ssh/id_rsa`
 - enter a new passphrase twice, which can be something really simple, you will need to remember it to use it in a few steps time. Note the command line remains empty as you type the password.
- authorise the use of the key by adding it to their trusted keys `$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys`
- alter SSH settings to read-only with the following settings
 - `$ chmod 700 ~/.ssh`
 - `$ chmod 600 ~/.ssh/*`
- Connect to wilson `$ ssh wilson`
- enter ssh passphrase created in earlier step, note this is NOT your fedID password
- if all has gone well then you should see this screen :

[illegible]

For Help or Support - Visit: <https://schelpdesk.diamond.ac.uk>

For jobs needing to access the GPFS02 filesystem please use the cs04r partition, for the GPFS03 filesystem please use the cs05r partition

Last login: Tue Jun 27 13:17:12 2023 from i07-ws004.diamond.ac.uk

```
[rpy65944@cs04r-sc-vserv-300 ~]$
```

- Check you can connect without needing password - type `$ exit`
- Type `$ ssh wilson`
- This should go straight to the welcome text above, without the need to input a passphrase
- Once this is successfully connecting to the wilson using SSH you are ready to follow the steps in the next 'Using fast_rsm' section

step-by-step instructions on how to use *fast_rsm* at i07 are detailed below. If you are entirely unfamiliar with linux/computers and you can't follow along, ask a beamline member of staff or email philip.mousley@diamond.ac.uk

1. Login to a diamond linux machine. If you're on the beamline, the beamline scientists will show you which machine you're on. If you're at home, use nomachine [<https://www.diamond.ac.uk/Users/Experiment-at-Diamond/IT-User-Guide/Not-at-DLS/Nomachine.html>] to access a linux workstation with access to the diamond servers.
2. type "module load fast_rsm" into the terminal
3. If this is your first time using the fast_rsm mapper, type "firstrun" to setup a folder in your home directory
4. type "activate" into the terminal to activate the fast_rsm conda environment
5. if you need a mask file for your data, make a mask by typing "makemask -dir path/to/experiment/directory -s ##scan##number#" to open up the mask GUI. Save the created mask and note down the file path ####/####/#####.edf
6. if you do not have an experiment file, make an experiment setup file by typing "makesetup" to open up a template experimental file, edit with your experimental information. this will include
 - a. *edfmaskfile* = path to the mask you just created.
 - b. *local_output_path*
 - c. *local_data_path*
 - d. *beam_centre*
 - e. *detector_distance*
 - f. *ensure your process_outputs are correct - explained lower down*
7. Then save this exp_setup.py file noting the path
8. run the processing by typing "process_scans -exp 'path/to/exp_setup.py' -s scan-numbers-to-be-mapped
 - a. e.g. process_scans -exp /home/rpy65944/fast_rsm/example_exp_setup.py -s 441187 441188
9. alternatively use the -sr option to define an evenly spaced range of scans using the format [start,stop,stepsize]
 - a. -sr [441187, 441189,1]
10. use a list of lists to define several sets of evenly spaced scans using the format [[start1,stop1,stepsize1],[start2,stop2,stepsize2]], where the ranges are inclusive i.e. the stop value is the final scan in the range which you want analysed
 - a. -sr [[441187, 441189,1],[441192, 441195,1]]

Job submitted, waiting for SLURM output. Timer=## --> *this line checks for a new SLURM output file every 5 seconds, current time limit of 250 seconds*

STARTING TO MONITOR TAIL END OF FILE, TO EXIT THIS VIEW PRESS ANY LETTER FOLLOWED BY ENTER --> monitoring of tail end of
slurm out file to monitor progress of calculation

Process_outputs explained

'full_reciprocal_map'

calculates a full reciprocal space map combining all scans listed into a single volume.

'curved_projection_2D'

this projects a series of detector images into a single 2D image, treating the images as if there were all from a curved detector. This will give a projected 2d image. This should only be used when a detector has been moved on the diffractometer arm during a scan, and the images need to be combined together. *NOTE: this will not work on a continuous scans with ~1000s of images - for these scan types use the 'large_moving_det' option.*

'pyfai_1D'

Does an azimuthal integration on an image using PONI and MASK settings described in corresponding files. This can handle a short series of images (~50) for individual integrations. If used in combination with 'curved_projection_2D', this will simply integrate the large projected image, and not the series of small images.

'qperp_qpara_map'

projects GIWAXS image into q_para,q_perp plot. NOTE: Similar to 'curved_projection_2D', this will not work with 1000's of images - for these scans use the 'large_moving_det' option.

'large_moving_det'

utilise MultiGeometry option in pyFAI for scan with a moving detector and a large number of images (~1000s), outputs: I, Q, two theta, caked image,Q_para Vs Q_perp

other functions:

get_setups - use this to get setup information used to calculate a hdf5 file. if hdf5 has the configuration saved to config, and it contains the joblines from the analysis job sent to the cluster - then you can type

```
get_setups -hf path/to/hdf5file.hdf5 -outdir path/to/out/directory
```

This will open up the hdf5 file, locate the joblines information, and then it will save a copy of this information and open exp_setup.py and calc_setup.py files with the information that was used to send off the analysis job.

Debugging

1. If you can't get nomachine working, contact the beamline staff/diamond IT support.
2. -
3. If module load fast_rsm doesn't work:
 - a. If you get "command not found: module", you're probably on your own computer, not the diamond servers!
 - b. If you get "Unable to locate a modulefile for 'fast_rsm', there is a serious problem. Contact beamline staff immediately.
4. The slurmout file should contain error messages with information about what error was encountered. If these do not provide an easy solution, forward this onto the i07's data analysis scientist for support

Q: I finished processing, but the output looks ridiculous. What do I do?

There are many reasons why this could happen, but first make sure that you remembered to correctly set your central pixel (this is the easiest thing to forget). Otherwise, either some other piece of information has been entered incorrectly, or your data file is corrupt in some way. In case of the latter, contact the i07's data analysis scientist to investigate. It is usually possible, with some work, to recover your data.

map_per_image

There is a very important switch that changes the operating mode of fast_rsm: map_per_image. map_per_image defaults to False, in which case all of the scans will be combined into a single volume in reciprocal space. If map_per_image is set to true, then **a very large amount of information will be saved per image**. Generally speaking, if map_per_image is set to true and you are mapping 1GB of images, running the fast_rsm notebook could generate around 20GB of data.

If `map_per_image = True`, then, for each image in the scans selected, the following will be saved:

- A binned reciprocal space map, of exactly the same form as those generated when `map_per_image` is false. Each binned reciprocal space map will have the size set by
- ***Every q-vector for every pixel in the image.***
- Every intensity obtained for every pixel in the image, **without** polarisation correction being applied.
- Every intensity obtained for every pixel in the image, **with** polarisation correction applied.

Because `map_per_image` stores every q-vector for every pixel in the image, the complete scattering information is saved. From this, exact maps to e.g. $I(Q)$ and Q_{xy} - Q_z can be computed.