**Instructions for calibrating wavelength offsets and striping in MIRI data**

**Note on file names**

The band file names are assumed to be of the format

**Level3\_ch[channel]-[grating]\_s3d.fits**

eg

**Level3\_ch1-short\_s3d.fits**

**Wavelength offsets**

Create a working directory to run the calibration from

**mkdir [work\_dir]**

Copy the directory containing the data (**[uncal\_dir]**) to be calibrated into **[work\_dir]**

This directory should contain the band files in the following format

**[exposure]/[tile]/stage3/[dither]/stage3/[file\_name].fits**

eg

**july/centre/stage3/d1/stage3/Level3\_ch1-short\_s3d.fits**

If there are level 3 files generated with fewer groups (using **remove\_groups.py**) then these should be stored in the following format (where **number** is the number of groups used)

**[exposure]/[tile]/desaturation/group[number]/stage3/[dither]/stage3/[file\_name].fits**

eg

**july/centre/desaturation/group1/stage3/d1/stage3/Level3\_ch1-short\_s3d.fits**

The following directories and scripts are required within **[work\_dir]**

**copy\_cal.py**

**extract\_calibration.py**

**correct\_spaxel.py**

**mre\_files/**

**submit\_caljob**

**apply\_calibration.py**

These can be found under:

**/data/nemesis/jwst/scripts/calibrate\_miri**

on ALICE2 (University of Leicester users only)

Or on Github **Jharkett/MIRI-code** or **JWSTGiantPlanets/MIRI-Toolkit**

**mre\_files/** should contain files for each band of the format

**wavelength retrieved NEMESIS spectrum (µW cm-2 sr-1 µm-1)**

And named

**[band].txt**

eg

**ch1-short.txt**

An example of this mre directory used for GTO 1246 data can be seen under **/data/nemesis/jwst/scripts/calibrate\_miri/mre\_files\_grs**

This mre data is what the cube data is calibrated to, so make sure it is as similar to the cube data as possible (without the wavelength offsets). The closer it is, the better the calibration will be.

Firstly, a new directory **cal\_cubes/** should be created that contains the files (desaturated or not) that will be used to determine the required calibration for each band. This directory should be of the format

**[exposure]/[tile]/stage3/[dither]/[file\_name].fits**

eg

**july/centre/stage3/d1/Level3\_ch1-short\_s3d.fits**

This is done using **copy\_cal.py**

Open **copy\_cal.py**

line 14: input uncal dir

line 15: input the observation exposures (epochs)

line 16: input the tiles

line 18: input the bands to use desaturated data for calibration

line 19: input the number of groups to use for each of the above desaturated bands

Run **copy\_cal.py**

**python copy\_cal.py**

Open **extract\_calibration.py**

line 18: Input **[uncal\_dir]**

line 20: input exposures

line 21: Input tiles directory structure

line 22: Input band to be calibrated – of the format **ch[channel]-[grating]** eg: **ch2-short**

line 24: Input minimum wavelength to use (set outside the wavelength range to use lowest)

line 25: Input maximum wavelength to use (set outside the wavelength range to use highest)

line 27: Input minimum wavelength to apply the calibration to (calibration = 0 lower than this)

line 28: Input maximum wavelength to apply the calibration to (calibration = 0 higher than this)

line 27: Set threshold for allowed number of 0 values in a spectra (leave as 10)

Run **extract\_calibration.py**

Takes a few hours to run

When it is finished, consider plotting a handful of spectra to check they have been correctly calibrated

A new file **cal\_file.txt** is generated containing the inputs to **extract\_calibration.py**

These are used by the other wave-calibration scripts

Open **apply\_calibration.py**

line 23: Input name of output directory (will be of the form ‘input dir’ + **[…]**)

line 26: Only set to False if the wave\_cal file containing the wave correction for each pixel has already been made

line 31: Set to True if a change in the wave grid is required

line 32: If line 31 = True: set the incremental step for this new wave grid

line 34: Set to True if removal (sets to 0) of bad outlier pixels are required

line 35: Input (in the form of an array), the x positions of each bad pixel to remove

line 36: Input (in the form of an array), the y positions of each bad pixel to remove

Run **apply\_calibration.py**

**python apply\_calibration.py**

Produces a new file in directory **wave\_cal\_data/** that contains the calibration to be applied to each spaxel.

The calibrated data will be saved in a new directory **[uncal\_dir] + […]**

**Flat field generation**

The following scripts are required

**gen\_2d\_flat.py**

**apply\_2d\_flat.py**

These can be found under:

**/data/nemesis/jwst/scripts/calibrate\_miri**

on ALICE2 (University of Leicester users only)

Or on Github **Jharkett/MIRI-code** or **JWSTGiantPlanets/MIRI-Toolkit**

Navigate to either **[uncal\_dir]** (if wave-cal offsets are not being corrected) or **[uncal\_dir]\_cal**

Open **gen\_2d\_flat.py**

line 14: input the name of the band file to calibrate

line 15: input directory structure of exposures and tiles (note the user can use desaturated data for this too)

Run **gen\_2d\_flat.py**

A new directory **flat\_files/** will be produced containing the flat file for the chosen band made using all available tiles

Open **apply\_2d\_flat.py**

line 13: input directory structure of exposures and tiles

line 14: input name of stage3 directory where the calibrated data should be stored

line 15: input name of the flat file (default is **flat\_all\_epoch.fits**)

line 16: input the name of the band file to calibrate

Run  **apply\_2d\_flat.py**

Flat-fielded files will be generated in the input directory on line 14 of **apply\_2d\_flat.py**