# Guidelines and troubleshooting tips for MINDS Data Reduction Notebook v007

## 1. How to download the data and where to place them?

- The recommended way to download uncalibrated data (suffix: *uncal.fits) is with https://github.com/spacetelescope/jwst_mast_query. This involves generating a MAST token (https://auth.mast.stsci.edu/token) and setting the MAST_API_TOKEN environment variable as described here. Then adapt a config file with your favorite text editor in order to set what you want to download (example config file here), and run the download with:

  jwst_download.py -v --config jwst_query.cfg

- Create a subfolder named *stage0* in the folder you want to use as working directory during the reduction with the MINDS notebook (or adapt the parameter *input_dir* in cell [2] of the notebook accordingly).
- In the *stage0* subfolder, **only** place *uncal.fits from your science target. Check that the total number of files is a multiple of 12 (if 2-dither mode) or 24 (if 4-dither mode). If not the case, this means there are odd ones out, and **the notebook will fail**. Check carefully the headers and/or filenames to identify inappropriate input files (e.g. dedicated BKG observations).
- If you have dedicated background observations, place the associated *uncal.fits files in a different subfolder and adapt the *input_bgdir* and *output_bgdir* parameters in the MINDS notebook accordingly.

## 2. How to use the notebook (TLDR version)?

- Follow the **flow charts**:
  - Adapt paths in cell [2] to point to the data on your machine, and follow flow chart #1 to set other crucial parameters in that cell.
  - Do not change anything else and run the whole notebook.
  - Check your results in the *final_outputs* folder (details in Sec. 4).
  - Based on the results, follow flow chart #2 to potentially adapt parameters and run again until satisfied.

## 3. How to set free parameters in the MINDS notebook?

- The data reduction parameters that you're the most likely to adapt are in cell [2], and are detailed below.
- Parameters in cell [4] were once free parameters that have now been extensively tested. You most probably won't need to change them, except for very peculiar datasets.
- Directories:
  - Adapt `workdir` according to where you placed your uncalibrated data.
  - `Input_dir`, `output_dir` and `final_outdir` can be left as default.

    - If you have dedicated background observations, adapt `input_bgdir` and `output_bgdir` accordingly.
    - Adapt `source` accordingly (just used in output plot and file names).
- Which sections of the notebook to run (again)?
    - For the first run, set all `do_sec`*`m`* and `overwrite_`*`mn`* flags to True.
    - Then depending on the parts of the reduction you'd like to run again (for safety checks, or to test other parameters), adapt these values accordingly. To help you with setting the `overwrite` flags, parameters in cell [2] are sorted in different categories based on the Sections where they are first used (given in comments).
        - Example: You ran the notebook with bkg_method = 'annulus' and you now want to try a different background subtraction (with all else identical). Set `do_sec3=True, overwrite_31=False, overwrite_32=False, overwrite_33=True,` and all other flags to `True` (as the background subtraction occurs in Sec. 3.3, and would affect all files produced in subsequent Secs).
- Multiprocessing:
    - Adapt depending on whether you want to use a fraction of all available CPUs (`usage`), or a specific number of CPUs of your machine (`maxp`).
- Background subtraction
    - `bkg_method`:
        - By default, use 'annulus' (i.e. removes median pixel intensity estimated in an annulus around the aperture used for flux integration)
        - If the source is faint OR dedicated BKG observations are available: use BKG_method = 'ddither' (i.e. direct dither subtraction or direct BKG subtraction).
        - If your source is bright or somewhat extended (but you do not have dedicated BKG observations), you can try 'sdither' (BKG proxy will be the minimum of doublet/quadruplet of dither images after median-box smoothing).
    - `bg_observation`:
        - True if you have either a dedicated background observation or the observation of a faint star you would like to use as BKG proxy, False otherwise.
    - `dedicated_bg`:
        - True if it is a dedicated background observation, False if it is the observation of a faint star, used as BKG proxy.
    - `rm_persist`:
        - Whether to attempt removing with a simplistic assumption (controlled with parameter persist_frac) - most useful if your target is bright and extended.
- `psff`?
    - Leave to `False` for now.
    - This option, if set to `True,` allows to use point-source specific reference files from Danny Gasman. This requires that target acquisition was ON for your

dataset and that the source is unresolved (point-like). However, the current time correction in PHOTOM files distributed with the repository are inappropriate. The relevant files may or may not be updated in the future.

- Spectrum extraction
  - `apsize` [only used if psff=True] Size of the aperture, in FWHM, used for spectrum extraction. Default value (2) is the one used by the official pipeline.
  - `overwrite_target_classification` : Whether to overwrite the target classification. If psff=False, this can change the behaviour of the spectrum extraction. By default, if a source was labelled 'EXTENDED', it will extract the spectrum by summing all pixels in the field. Overwriting it to 'POINT' would make the extraction happen in a 2-FWHM aperture.
  - `new_sourcetype` : the new target classification if the above is set to True.
  - `spike_filtering` [only used if psff=True] Whether to attempt to filter out spikes from your spectrum. These are identified as individual outliers in the aperture used for spectrum extraction
  - `sp_sig` [only used if `spike_filtering=True`] Sigma used to identify spikes by sigma filtering of individual spaxels in the extraction aperture.
  - `max_nspax` [only used if `spike_filtering=True`] Maximum number of spaxels identified as spectral outliers at the same wavelength, for a given pixel at a given wavelength to be considered a spike. For example: if set to 2, only one other pixel can be identified as bad. If more are found, the algorithm will assume there is a real high-spectral frequency feature (e.g. a thin absorption/emission line) and will leave all pixels as they are.
  - `neg_only` [only used if `spike_filtering=True`] : whether to correct only for negative spikes.
- Operations on spectra
  - `do_rfc1d` : Whether to apply 1D residual fringe correction (leveraging the jwst pipeline implementation made by Patrick Kavanagh).
  - `nch_bad_per_band` :Number of spectral slices to discard at the very beginning and very end of each band. This should be a list of 12 elements corresponding to the 12 bands, in increasing wavelength order. If an element is an integer, that number of slices is discarded from both the beginning and end of the band. If an element is a tuple/list of 2 integers, these will correspond to the number of slices to be removed at the beginning and end of the band, respectively.
  - `correct_jumps` : if not None, the method to stitch bands together:
    - 'scale_only': allows scaling of bands such that their shortward section have a similar level as the overlapping longward section of previous band;
    - 'occam': for each band, either (i) 'scale_only' above, or (ii) correct for slope of the band to match neighbouring band levels. Default behaviour is 'scale_only', EXCEPT if this leads to all subsequent bands to require a similar scaling (e.g. third band requires 1.10 factor to match second band, BUT all subsequent bands also require ~1.10 factor => Occam Razor: it's

more likely that the slope of third band is wrong, while all subsequent bands could be well calibrated).

- Plots
  - `show_plots` : Whether to show all plots
  - `wl_min, wl_max` : Min and max wavelengths of the plotted spectra.
  - `spitzer_file` : If not None: full path + filename of another spectrum - e.g. a Spitzer spectrum. The file needs to be parsable by pandas.read_csv, and contain columns labelled "lambda", "Flux" (in Jy), and 'F_Er' (in Jy).
  - `photom_file` : if not None, as above: full path + filename of another SED - e.g. Photometry points. The file needs to be parsable by pandas.read_csv, and contain columns labelled "lambda", "Flux" (in Jy), and 'F_Er' (in Jy).
  - `compare_spec` : whether to compare spectra obtained with different reduction parameters (these are found automatically in intermediate stage3_007b folder).
  - `make_aperture_image` : whether to create a figure showing the aperture size with respect to a spectral image at a given wavelength (as done in the PDS 70 paper Perotti et al. 2023)

**4. Where are my results? What are all my outputs?**

- By default, intermediate outputs will be written in *stage1*, *stage2*, and *stage3* subfolders. These can be used to check that the different steps of the notebook do what they should on your data, troubleshooting or debugging (see troubleshooting section in some ).
- A *final_outputs* subfolder will also be created, and will contain the final results you are likely interested in:
  - A *figures* subsubfolder containing:
    - Different spectra plots, for a first quick look.
    - If requested, the aperture used for spectrum extraction plotted on top images at selected wavelengths.
  - A *stage3_00n* subsubfolder (*n* being the MINDS notebook version used) with:
    - *cxy_med.fits* and *cxy_med_ifua.fits*: The star centroid (x,y) coordinates in each of the 12 bands (used as aperture centers), for sky-aligned and ifu-aligned results, respectively.
    - Final spectral cubes (one for each of the 12 bands), recentered if cen_method != None. Suffix: *s3d_cen.fits*
    - Final collated spectra in a **csv table** with different columns. Suffix: .csv
      - Column “wavelength”: Wavelengths
      - Column “flux”: Corresponding fluxes without band stitching nor residual fringe correction.
      - Column “flux_error”: Uncertainty on the flux, as returned by the JWST pipeline (as of v1.11, it only captures Poisson noise). If you selected `bkg_method='sdither'` and `psff=True`, the uncertainty also encompasses the uncertainty on the background flux.

        - [if `do_rfc1d=True`] Column “flux_fcorr”: fluxes after residual fringe correction.
        - [if `correct_jumps='scale_only'/'occam'`] Column “flux_stitch_scale_only” or “flux_stitch_occam”: fluxes after band stitching. Note: if `do_rfc1d=True`, you also get a “flux_stitch_scale_only_fcorr” / “flux_stitch_occam_fcorr” column, which combines both residual fringe correction and stitching.
    - Final extracted spectra per band in either .fits or .txt format. Suffix: *x1d*.fits* and *x1d*.txt*, where the * can replace ‘_minds’ or ‘vanilla’, for the MINDS notebook extraction with chosen parameters, or the MINDS notebook extraction with pipeline default spectrum extraction parameters - i.e. still leverages the BKG extraction from MINDS notebook.

**5. Are my results reliable? Safety checks.**

- Each time you want to run the notebook with a different reduction parameter, make sure to adapt the overwrite flags wisely.
- In case no dedicated BKG observation is available, it is recommended to run the reduction with the 3 different *BKG_method* available, and compare the final spectra to assess which yields the best results (quality of the spectrum at different wavelengths depends on source brightness and spectrum, and whether it is spatially extended)
- If you want to permanently save some results, but test how results change with other parameters, simply change the *final_outdir* parameter before launching a reduction with other parameters.

**6. FAQ**

- Problem: I experience repeated disconnections from the jupyter notebook. What can I do?
    - Solution: `pip install --upgrade ipyflow jupyter_client jupyter_server jupyter_core tornado`
- Problem: Det1 (or Spec2) pipeline appears to get stuck without doing anything (Sec. 3.1 or Sec. 3.2.), and without showing any error message.
    - Solution(s):
        - Be patient, depending on the specs of your machine and/or available memory it may take some time (sometimes up to ~20min) for a new output to be printed out.
        - If it is the first time you are running the new pipeline version or use the latest reference files, they may have to be downloaded from CRDS and the calculation time will be limited by your internet bandwidth.

        - Have you adapted the number of requested CPUs in the second cell to match the ones available on your machine? By default `maxp=4`. If too much for your machine, it can increase calculation times.
- Problem: Det1 pipeline shows error messages like: `ERROR - Error determining best reference for 'pars-jumpstep' = parameter='META.INSTRUMENT...'` or `ERROR - Error determining best reference for 'pars-undersamplingcorrectionstep' = Unknown reference type 'pars…'`
    - Solution: this is fine if a message including `INFO - PARS-DETECTOR1PIPELINE parameters found: </local/path/to/some/asdf/file>` is displayed just after the error messages.
- Problem: My spectrum looks very strange in some bands, with large offsets from one band to the next.
    - Is your source very faint? Check the inferred center values. If the source is too faint in some bands, the algorithm may not find its centroid automatically. Solution: Check manually the final *s3d_cen.fits cubes, note in which bands you do not see visually any source in the mean image of the band (if you don't see it the algorithm likely won't see it either), and use the discard_band_cen parameter to provide a list of these bands where centering should be avoided.
    - If the source is too faint beyond a certain wavelength, adapt the `wl_max` parameter (Sec. 6) correspondingly to adapt the wavelength range of your final plots.
    - Is your source very close to the edge of the field? Different fractions of the PSF may be cropped out from one channel to the next. You may consider using the following rescaling options (with care), keeping in mind that.
- Problem: I suspect my spectrum to be plagued with a lot of **spikes**, which makes continuum identification difficult.
    - Try setting `spike_filtering=True`, re-run the notebook, and cross fingers. You can also provide a list of booleans of length 12, corresponding to the bands where you want the spike filtering to occur.
- Problem: I requested stitching (`correct_jumps != None`) or residual fringe correction (`do_rfc1d=True`), but upon plotting the final spectra from the spectrum_vanilla.csv or spectrum_minds.csv files, it doesn't look like these corrections were made:
    - spectrum_vanilla.csv: does not contain (it aims to reproduce the closest spectra as could be obtained with the JWST pipeline alone). 'flux' and 'flux_fcorr' will contain the spectra without and with residual fringe correction, respectively (see Sec. 4).
    - spectrum_minds.csv: you'll need to look at the appropriate column of the csv file, whose exact name will depend on the values adopted for `correct_jumps` and `do_rfc1d` (see Sec. 4). The 'flux' column only contains the spectrum without stitching nor residual fringe correction.

- Problem: I get a `FileNotFoundError: [Errno 2] No such file or directory: '<path>/<some_filename>.fits.fits'`
    - Most likely issue is that you changed some paths and may have not turned on again the relevant do_sec* flags.
    - The ".fits.fits" part is normal. The fits utility function that the pipeline leverages first tries to find a given filename, but if it can't, it tries as second resort to add a ".fits" as suffix (which can mean a double ".fits"), then raises an error if it still can't find the latter.