Manual
for
NIRCam Coronagraphy Simulations

Jens Kammerer, Julien Girard, Jarron Leisenring, Ben Sunnquist, Marshall Perrin

Table of Contents

Limitations

- In its current version, the pipeline can only be used to simulate programs consisting of one or multiple **roll 1, roll 2, reference observation sequences**. Also, all observations will be simulated at the same observation date.
- **Standard subpixel dithering** is not supported and **small grid dithering** is only supported for reference observations, but not for science observations.
- The pipeline has been tested with the **GTO programs 1194** (HR 8799), **1411** (beta Pic), and **1412** (51 Eri). The corresponding configuration files are provided as `hr8799.yaml`, `betapic.yaml`, and `51eri.yaml` and the corresponding APT files are provided in the HR8799, betaPic, and 51Eri folders. For testing the pipeline with these example programs, copy and rename the desired configuration file to `config.yaml`.

History

- Version 2.0.0
  - INSTALLATION OF NEW VERSION OF pyNRC IS REQUIRED!
    - See "Installation instructions/New pyNRC environment".
  - There is a new script `run_pynrc_new.py` that can be used to simulate pipeline-ready ramp and quick-look slope images using the new version of pyNRC.
  - Target acquisition and pointing errors are now simulated as well as cosmic rays.
  - Target Acquisition and Astrometric Confirmation images are now simulated.
  - The stage 3 pipeline is now working correctly.
- Version 1.0.0
  - UPDATE OF pyNRC IS REQUIRED!
  - The jump detection threshold was increased to 50 because otherwise too many pixels were flagged as bad in the coronagraphic subarrays.
  - The co-adding step of the stage 3 pipeline was improved by hard-coding the use of outdated distortion reference files which correspond to the tweaked reference pixel positions (see version 0.4.0).
  - Simulations with the true reference pixel positions and the most recent distortion reference files can be obtained by commenting in/out the corresponding lines in `util.py` and `run_jwst_s1s2.py`, but the stage 3 co-adding step will not work correctly anymore.
- Version 0.4.0
  - The issue with mismatched bad pixel maps was fixed. It was caused by an offset between the reference files used by pyNRC and those on CRDS.
  - The co-adding step of the stage 3 pipeline was improved by tweaking the reference pixel positions.
- Version 0.3.0
  - The JWST stage 3 pipeline processing was improved by fixing multiple wrong header keywords.
  - The issue with the wrongly oriented small grid dithers was fixed by inverting the RA sign of the dither offset.
  - The issue with the axis flip between pyNRC and MIRAGE data was fixed for both channels.
  - The issue with the axis flip of the coronagraphic masks was fixed for both channels.

- Version 0.2.0
  - The issue with the wrongly oriented scenes was fixed by injecting the companions with an inverted RA sign in `run_pynrc.py`.
  - The issue with the horizontal axis flip between pyNRC and MIRAGE data was fixed for the LW channel by flipping the CRDS reference files read by pyNRC.
  - The issue with the horizontal axis flip of the coronagraphic masks was fixed for the LW channel by flipping the masks in pyNRC.
  - The issue with the large amount of bad pixels after the JWST stage 1 pipeline processing was fixed by skipping the dark subtraction step, i.e., setting `result1.dark_current.skip = True` in `run_jwst_s1s2.py`.
  - The issue with the slightly misaligned JWST stage 3 psfalign pipeline product was fixed by explicitly specifying the bad pixel types in the PSF alignment step, i.e., setting `result3.align_refs.bad_bits = 'HOT, UNRELIABLE_BIAS'` in `run_jwst_s3.py`.
- Version 0.1.0
  - First commit.

Installation instructions

The pipeline is separated into multiple Python scripts which need to be run one after another. The reason for this approach is that a different Python environment should be used for each of the software packages that are used by the pipeline. In the following, installation instructions for each of the different Python environments are given.

species environment

1) Create a new species Python environment.
2) Use `git clone https://github.com/tomasstolker/species.git` followed by `git reset --hard 165554cde2ad65abc8501c11c188258264d7abfb` to clone **version 0.4.0** of species into a directory of your choice. Specify this directory under `species_dir` in the configuration file.
3) Install the required dependencies into the species Python environment.
4) Note that an installation of species itself, e.g., by running `setup.py`, is not recommended. Instead, just clone the repository and try running the `run_species.py` script. Python will crash and complain about the missing dependencies which you can then install one after another before trying it again.

pyNRC environment

If you want to use the new version of pyNRC to simulate your data, please skip this step and go to "New pyNRC environment"!

1) Create a new pyNRC Python environment.
2) Install the dependencies and data files required by pyNRC following a-g below. Further information can be found at https://github.com/kammerje/pynrc/blob/kammerje-patch-1/docs/install_clean.rst. Note that steps d and e are not mentioned in this documentation, but are required to run my modified version of pyNRC.
    a. Install Pysynphot, download the Pysynphot data files, untar them, and set the environment variable PYSYN_CDBS in your `.bashrc` file to point to them (`export PYSYN_CDBS='$HOME/data/cdbs/'`).
    b. Install jwxml and WebbPSF, download the WebbPSF data files, untar them, and set the environment variable WEBBPSF_PATH in your `.bashrc` file to point to them (`export WEBBPSF_PATH='$HOME/data/webbpsf-data/'`).
    c. Install JWST Backgrounds and its dependencies.
    d. Install WebbPSF_ext using `git clone https://github.com/JarronL/webbpsf_ext.git` followed by `git reset --hard 19df5301f0062a3fd56a38b05ad325cdb791fafe` to clone the correct version of WebbPSF_ext into a directory of your choice. Specify this directory under `webbpsf_ext_dir` in the configuration file. Run `setup.py` to complete the integration of WebbPSF_ext into WebbPSF.
    e. Set the environment variable WEBBPSF_EXT_PATH in your `.bashrc` file to point to the WebbPSF data files (`export WEBBPSF_EXT_PATH='$HOME/data/webbpsf-data/'`).
    f. Install pyNRC using `git clone https://github.com/kammerje/pynrc.git --branch kammerje-patch-1 --single-branch` to clone my modified version of pyNRC into a directory of your choice. Specify this directory under `pynrc_dir` in the configuration file.

g. Download the pyNRC data files, untar them, and set the environment variable PYNRC_PATH in your `.bashrc` file to point to them (`export PYNRC_PATH='$HOME/data/pynrc_data/'`).

3) Use `git clone https://github.com/semaphoreP/whereistheplanet.git` to clone the latest version of whereistheplanet into a directory of your choice. Specify this directory under `whereistheplanet_dir` in the configuration file.
4) Install the required dependencies into the pyNRC Python environment.

New pyNRC environment

1) Create a new pyNRC_new Python environment.
2) Make a folder for all the required software `mkdir soft`.
3) Navigate into that folder `cd soft/`.
4) Install pysiaf:
   a. `git clone https://github.com/spacetelescope/pysiaf`
   b. `cd pysiaf/`
   c. `pip install -e .`
5) Install poppy:
   a. `cd ..`
   b. `git clone https://github.com/spacetelescope/poppy`
   c. `cd poppy/`
   d. `pip install -e .`
6) Install webbpsf:
   a. `cd ..`
   b. `git clone https://github.com/spacetelescope/webbpsf`
   c. `cd webbpsf/`
   d. `pip install -e .`
7) Download webbpsf-data files:
   a. Download the latest webbpsf-data files into the webbpsf-data directory:
      i. https://stsci.app.box.com/s/g8nq6j1lk10ml7tyxxn28rbl40h1zky2
   b. Set the environment variable $WEBBPSF_PATH to point to that directory:
      i. In .bashrc add: `export WEBBPSF_PATH='$HOME/soft/webbpsf-data/`
8) Install jwst_backgrounds:
   a. `cd ..`
   b. `git clone https://github.com/spacetelescope/jwst_backgrounds`
   c. `cd jwst_backgrounds/`
   d. `pip install -e .`
9) Install jwst pipeline:
   a. `cd ..`
   b. `git clone https://github.com/spacetelescope/stcal`
   c. `cd stcal/`
   d. `pip install -e .`
   e. `cd ..`
   f. `git clone https://github.com/spacetelescope/stdatamodels`
   g. `cd stdatamodels/`
   h. `pip install -e .`
   i. `cd ..`
   j. `git clone https://github.com/spacetelescope/crds`
   k. `cd crds/`
   l. `pip install -e .`
   m. `cd ..`
   n. `git clone https://github.com/spacetelescope/stpipe`
   o. `cd stpipe/`
   p. `pip install -e .`
   q. `cd ..`

   r. `git clone https://github.com/spacetelescope/jwst`
   s. `cd jwst/`
   t. `pip install -e .`

10) Configure CRDS environment:
   a. In .bashrc add: `export CRDS_PATH='$HOME/data/crds_cache/'` (you can specify a directory of your choice)
   b. In .bashrc add: `export CRDS_SERVER_URL='https://jwst-crds.stsci.edu'`
11) Install pysynphot:
   a. conda install pysynphot
   b. Download and untar cdbs files:
      i. http://mips.as.arizona.edu/~jleisenring/pynrc/cdbs.tar.gz
   c. Set the environment variable $PYSYN_CDBS to point to that directory:
      i. In .bashrc add: `export PYSYN_CDBS='$HOME/data/cdbs/`
12) Install webbpsf_ext:
   a. `cd ..`
   b. `git clone https://github.com/JarronL/webbpsf_ext`
   c. `cd webbpsf_ext/`
   d. `pip install -e .`
13) Install pynrc:
   a. `cd ..`
   b. `git clone https://github.com/JarronL/pynrc`
   c. `cd pynrc/`
   d. `pip install -e .`
14) Download pynrc_data files:
   a. Download the latest pynrc_data files into the pynrc_data directory:
      i. http://mips.as.arizona.edu/~jleisenring/pynrc/
   b. Set the environment variable $PYNRC_DATA to point to that directory:
      i. In .bashrc add: `export PYNRC_DATA='$HOME/data/pynrc_data/'`
15) Make sure that you have:
   a. Version 2.8.3 of asdf

MIRAGE environment

If you want to use the new version of pyNRC to simulate your data, please skip this step!

1) Create a new MIRAGE Python environment.
2) Install **version 2.1.0** of MIRAGE from Pypi following https://mirage-data-simulator.readthedocs.io/en/latest/install.html#install-from-pypi.
3) If you encounter issues while running the MIRAGE scripts, make sure to have version 1.15.0 of healpy, version 2.1.0 of mirage, version 1.32 of grismconf, version 1.58 of nircam-gsim, and version 1.3.3 of jwst installed.
4) The required reference files will be downloaded using the `ref_mirage.py` Python script later.

JWST environment

1) Create a new JWST Python environment.
2) Install **version 1.3.3** of the JWST data reduction pipeline from Pypi following https://jwst-pipeline.readthedocs.io/en/latest/index.html.

Preparations

Before the pipeline can be run, several preparations need to be conducted. These involve modifying the APT file, modifying the configuration file, and downloading the MIRAGE reference files (the latter needs to be done only once before the pipeline is run for the first time).

APT file

If you want to use the new version of pyNRC to simulate your data, you also need the .smart_accounting and .timing.json files from APT. Contact Jarron or myself to help you with that.

1) Open the program with the NIRCam coronagraphy observations for which data shall be simulated in APT.
2) From the `Observations` folder of that program, remove all non-NIRCam non-coronagraphic observations.
3) Re-run the `Visit Planner`.
4) Under `Reports` → `Visit X:X` → `Total Roll Analysis For Visit`, find the preferred observation date and the corresponding roll angle constraints. These need to be specified under `date`, `pa_roll1`, `pa_roll2`, and `pa_ref` in the configuration file.
5) Save the xml and pointing files of the modified program using `File` → `Export…` → `xml file` & `pointing file` → `OK`.

Configuration file

1) **The pipeline always reads the parameters saved in the `config.yaml` file!**
2) In the `paths` section, once the directories of species, whereistheplanet, WebbPSF_ext, and pyNRC have been set correctly, only the `wdir` needs to be changed if simulations for a new program shall be made.
3) In the `apt` section, the paths of the xml and pointing files (relative to the `wdir`) need to be specified.
4) In the `observation` section, the observation date, roll angles, wavefront drifts, and oversampling need to be specified. Furthermore, the observing sequences need to be specified as `X,Y,Z`, where X/Y/Z are the observation indexes (starting with 1) of the roll 1/roll 2/reference observations in the APT file (these are not necessarily the observation numbers used in the APT file!).
5) In the `sources` section, the science and reference source names and properties need to be specified. The names must match the source names ("Name in the Proposal") used in the APT file.
6) If you want to use the new version of pyNRC to simulate your data, you also need to specify the name of a votable within the `wdir` under `vot`. The votable should contain stellar photometry from http://vizier.u-strasbg.fr/vizier/sed/.
7) In the `companions` section, an arbitrary number of companions c1-cX can be added. For each companion, besides a mass and a specific entropy at formation, a `name_witp` (name in whereistheplanet) needs to be specified. A list of available companions can be found at https://github.com/semaphoreP/whereistheplanet/blob/master/whereistheplanet/whereistheplanet.py. If a companion is not available in whereistheplanet, an arbitrary identifier (that does NOT match any of the available companions) needs to be used for `name_witp` and `ra_off` and `de_off` at the observation date need to be specified. Furthermore, a `name_spec` (name in species) needs to be specified. A list of available

companions can be found at https://github.com/tomasstolker/species/blob/master/species/data/companions.py . If a companion is not available in species, name_spec can be left blank and the companion magnitudes need to be computed in a custom way and saved as arrays of shape (1,) under name_filter.npy (where name = name_witp and filter = FXXXW/FXXXM/FXXXN) into the pmdir specified in the configuration file.

8) In the pipeline section, the model used by species to fit the observed companion photometry and its effective temperature range can be specified. A list of available models can be found at https://github.com/tomasstolker/species/blob/1dada33c9547f5bb2721687caf3c7d5d3d171856/species/data/database.py#L332. Furthermore, the make_plots parameter can be used to control whether plots shall be generated and saved into the pynrc_figs_dir.

MIRAGE reference files

1) Run the Python script ref_mirage.py in the MIRAGE Python environment, which will download the MIRAGE reference files into the mirage_refs_dir specified in the configuration file.
2) By default, this will only download a single linearized dark for each detector (total size of reference files ~95 GB). While it is recommended to download all linearized darks for better performance, this is irrelevant here since the ramp images in the MIRAGE data will be replaced with those from pyNRC eventually.
3) If you already have the MIRAGE reference files on your machine, you can simply skip this step. Just specify the mirage_refs_dir in the configuration file so that MIRAGE will be able to find them.

Running the pipeline

The pipeline is separated into multiple Python scripts which need to be run one after another. In summary, the pipeline computes the companion magnitudes in the relevant JWST bands using species, computes the companion locations at the specified observation date using whereistheplanet, simulates coronagraphic observations for the specified APT file using pyNRC, simulates the corresponding clear pupil observations using MIRAGE, replaces the ramp images in the MIRAGE data with those from the pyNRC data, and finally runs the simulated data through the JWST data reduction pipeline.

Running species

1) Run the Python script `run_species.py` in the species Python environment, which will compute the companion magnitudes in the relevant JWST filters and save them into the `pmdir` specified in the configuration file.
2) By default, the Exo-REM models with an effective temperature range of 1000-2000 K are used to fit the companion magnitudes from the literature. This can be changed using the `model_spec` and `teff_range` parameters in the configuration file. A list of available models can be found at https://github.com/tomasstolker/species/blob/1dada33c9547f5bb2721687caf3c7d5d3d171856/species/data/database.py#L332.
3) If desired (or if a companion is not available in species), the companion magnitudes can also be computed in a custom way and running `run_species.py` can be skipped. In that case, the `name_spec` parameter in the configuration file can be left blank and the companion magnitudes need to be saved as arrays of shape (1,) under `name_filter.npy` (where name = `name_witp` and filter = FXXXW/FXXXM/FXXXN) into the `pmdir` specified in the configuration file.

Running pyNRC

If you want to use the new version of pyNRC to simulate your data, skip all steps until "Running new version of pyNRC"!

1) Run the Python script `run_pynrc.py` in the pyNRC Python environment, which will simulate coronagraphic observations (ramps and noiseless slopes) for the specified APT file and save them into the `pynrc_data_dir` specified in the configuration file.
2) The `make_plots` parameter in the configuration file can be used to control whether plots shall be generated and saved into the `pynrc_figs_dir` specified in the configuration file.
3) The companion locations at the specified observation date are computed using whereistheplanet, but if a companion is not available in whereistheplanet, it is also possible to manually specify its `ra_off` and `de_off` in the configuration file. In that case, a `name_witp` still needs to be provided since this name is used as an identifier for the companion within the Python script.

Running MIRAGE

1) Run the Python script `run_mirage.py` in the MIRAGE Python environment, which will simulate clear pupil observations for the specified APT file and save them into the `mirage_data_dir` specified in the configuration file.
2) A random source at RA = DEC = 0 with a brightness of 20 mag in all JWST filters is used in this step. This should be irrelevant since the ramp images in the MIRAGE data will be replaced with those from pyNRC in the next step.

Replacing MIRAGE with pyNRC data

1) Run the Python script `run_pynrc_into_mirage.py` in the MIRAGE Python environment, which will replace the ramp images in the MIRAGE data with those from the pyNRC data and modify the relevant header keywords.

Running new version of pyNRC

1) Run the Python script `run_pynrc_new.py` in the pyNRC_new Python environment, which will simulate coronagraphic observations (ramps and noiseless slopes) for the specified APT file and save them into the `pynrc_data_dir` specified in the configuration file.
2) The companion locations at the specified observation date are computed using whereistheplanet, but if a companion is not available in whereistheplanet, it is also possible to manually specify its `ra_off` and `de_off` in the configuration file. In that case, a `name_witp` still needs to be provided since this name is used as an identifier for the companion within the Python script.

Running JWST data reduction pipeline

1) Run the Python script `run_jwst_s1s2.py` in the JWST Python environment, which will run the simulated ramp images through the JWST Stage 1 and 2 data reduction pipelines and save the output files into the `jwst_s1s2_data_dir` specified in the configuration file.
2) Run the Python script `run_jwst_s3.py` in the JWST Python environment, which will create the required ASN files, run the Stage 2 reduced data through the JWST Stage 3 data reduction pipeline and save the output files into the `jwst_s3_data_dir` specified in the configuration file.
3) **Note that the PSF alignment step can require a lot of memory allocation and processing time if the number of science integrations multiplied by the number of reference integrations is of the order of 1000 or larger.**