COMAP reduce docs]**COMAP Memo 0: COMAP Manchester Reduction Pipeline Documentation**

[

Stuart Harper

March 5, 2019

Contents

1 Overview **3**

2 Running the Pipeline **4**

2.1 The Code 4

2.2 Installation 4

2.2.1 General Linux Machine 4

2.3 Data Reduction Pipeline 5

2.3.1 Running the Pipeline 5

2.3.2 Parameter Files 5

2.3.3 Pipeline Overview 6

2.4 Running the Map-Maker 6

3 Old-Stuff **7**

4 Usage **8**

4.1 Parameter Files 8

5 Classes **8**

5.1 BaseClass.H5Data 8

1 Overview

The Manchester COMAP data reduction pipeline is designed to take the uncalibrated, full resolution COMAP *level 1* data and reduce it to wide 1 to 2 GHz bands that are calibrated using astronomical sources. We refer to the data calibrated using the *vane* and downsampled to 32 MHz channels as *level 2* data. After this any further reductions are called *level 3* data where astronomical calibration has been applied, and the data has been filtered of bad frequency channels and as many systematics as we can. *Level 3* data is intended to be passed directly to a map-maker. There may be several *level 3* datasets.

The final step of the Manchester COMAP pipeline is to filter out all observations with bad weather, poor noise statistics, or any other poor performance criteria. After the data have been filtered we can pass the seleceted good *level 3* data to the Destriping map-maker, resulting in a sky map, covariance map, and hit distribution map stored in `FITS` standard format with some predefined World Coordinate System (WCS).

A quick overview of the processing pipeline is:

1. All level 1 data is processed through several modules:
   - Measure the system temperature ($T_{\rm sys}$) and gain (G) using the calibration vane.
     [`Calibration.CalculateVaneMeasurement`]
   - Calibrate the *level1* data using the *vane*.
     [`Calibration.CreateLevel2Cont`]
   - Define edges of *scans* and store these.
     [`Statistics.ScanEdges`]
   - Flag out any large spikes in the data exceeding 10σ of the noise.
     [`Flagging.SigmaClip`]
   - Measure noise statistics per 32 MHz channel and fit 1/f statistics.
     [`Statistics.FnoiseStats`]
2. The resulting *level 2* data is then further processed into *level 3* data. There are several choices to make:
   - File lists of good data based on noise statistics.
   - Astronomical calibration to apply - leave vane calibration? Tau A? Jupiter?

   - Channel masks to apply, and account for how these change with time.
   - Finally choose run name and create *level 3* reduction. [`CreateLevel3.CreateLevel3`]

2 Running the Pipeline

2.1 The Code

All the code you need for the pipeline are stored in the `github` repository found here: `https://github.com/SharperJBCA/COMAPreduce`. There are three modules:

- `Analysis` - This contains all the data reduction pipeline for making *level 3* data.
- `MapMaking` - All the code for making maps.
- `Tools` - Extra scripts that may be useful like coordinate transforms, stats not included in scipy, etc...

2.2 Installation

2.2.1 General Linux Machine

First you will need to install `Python` 3.X. To do this I recommend download the Anaconda distribution. Go to: `https://www.anaconda.com/products/individual`. This will provide you with a `.sh` file that from the command line you can install by:

```
> bash <filename>.sh
```

From here just follow the instructions, and make sure you install Anaconda somewhere on the local disks (e.g., `/local/scratch/<username>`). Now you will have a new command line tool call `conda` and you will be using this to setup your environment and install new `Python` packages. The first thing you must do after install `conda` is link the conda-forge (`https://conda-forge.org/`) repository, to do this run the command:

```
> conda config --add channels conda-forge
> conda config --set channel_priority struct
```

Next you will want to install the pipeline code. First you must clone the `github` repo to somewhere on your machine:

```
> git clone https://github.com/SharperJBCA/COMAPreduce.git .
```

This will create a directory called COMAPreduce, enter this. We will now create a new virtual environment that matches mine so that you can be sure we are running the code with all the same packages. To do this simply execute:

```
> conda create --name <env> --file requirements.txt
```

Where `<env>` is the name of the virtual environment you want to use. Whenever you `ssh` into the machine you will need to use the command:

```
> conda activate <env>
```

to ensure your system is setup correctly. Alternatively you can add this line to your `rc` file (e.g., `.tcshrc` in your home area).

Finally, you can now install the pipeline routines as a module into your current `Python` setup by simply invoking:

```
> python setup.py install
```

in the COMAPreduce directory.

2.3 Data Reduction Pipeline

2.3.1 Running the Pipeline

a

```
> python run_average.py <parameter_file>
```

a

```
> mpirun -n X python batch_job.py <parameter_file>
```

a

```
> qsub pbs.script
```

2.3.2 Parameter Files

Jobs are defined by two parameter files. The primary parameter file defines the *pipeline* to be run, the location of a text file containing the data to be processed, and also the location of a *class-parameters* file that defines the inputs for each job.

The main parameter file is constructed like:

```
[Inputs]
pipeline:  Calibration.CalculateVaneMeasurement,...
classParameters = ParameterFiles/ClassParameters.ini
filelist = Filelists/CygA_output.list
LogFile = logs/logfile.txt
LogFileUsePID = True
```

2.3.3 Pipeline Overview

There are three main steps in the pipeline:

1. Vane Calibration:
   - Measure vane T_{sys}, gain and T_{sys} spikes (`CalculateVaneMeasurement`).
   - Create *level 2* data by down sampling in frequency and optionally applying the vane calibration. (`CreateLevel2Cont`).
2. Source Fitting:
   - If the observation is of jupiter/TauA/CygA/CasA fit the source brightness for later astronomical calibration. (`FitSource`).
3. Statistics:
   - Define the regions of an observation that are *on-source*, e.g. remove vane events and repointing events. (`ScanEdges`)
   - Flag time-ordered transients from the data using a sigma-clip after filtering large-scale correlated noise. (`SigmaClip`).
   - Measure the noise statistics, atmospheric fluctuations, and large-scale correlated noise for data quality purposes. (`FnoiseStats`).

2.4 Running the Map-Maker

To run the map-maker create a new working directory somewhere on your system. Next copy from the `COMAPreduce/comancpipeline/MapMaking` directory the following files/directorys to your working direcotry:

- `run_destriper.py`
- `CreateFilelist.py`
- `ParameterFiles`

Inside the `ParameterFiles` directory you will find a number of `.ini` files that are pre-setup for running the M31 data (fg4), the Galactic plane survey (GFields/fg6), and the Perseus data (fg7). These are called:

- `ParameterFilesparameters_fg4.ini`
- `ParameterFilesparameters_fg7.ini`
- `ParameterFilesparameters_GFields.ini`

First you will need to create filelists of processed *level 3* data files by invoking the `CreateFilelists.py` script:

```
> python CreateFilelists.py <source-name>
```

where `<source-name>` is fg4, fg7, GField, etc... Note: feel free to edit this file so you can optimise the file lists. The filelists will be stroed in a `FileLists/` directory.

Once you have a filelist ready you can invoke the map-maker using:

```
> python run_destriper.py <parameter-file>
```

3 Old-Stuff

Requirements:

- Python 3.0 or higher.
- Compiled shared libraries of the `FORTRAN` version of the Starlink astronomical libraries (SLALIB) available from http://starlink.eao.hawaii.edu/starlink/2018ADownload.
- A copy of parallel ready H5Py. Installation instructions can be found here http://docs.h5py.org/en/stable/mpi.html. N.B.: If you are using an Anaconda packaged version of Python installation you may need to remove the existing conda install of HDF5.
- The latest version of HealPy , mpi4py (either openMPI or MPICH work as backends), numpy, scipy, matplotlib, and astropy.

To install the Manchester COMAP reduction pipeline:

- Clone/download the `github` repository found here: https://github.com/SharperJBCA/COMAPreduce.

- Enter the directory: *cd COMAPreduce* and run *python setup.py install*. If your SLALIB libaries are not in standard location you must define the environment variable: `SLALIB_LIBS`

- To run the COMAP pipeline make a new directory above COMAPreduce (e.g. *cd ../ && mkdir runcomapreduce*) and copy the RUN.PY, and .INI files there.

- The pipeline can then be run using the command: *mpirun -n X python run.py -F FILELIST.list -P PARAMETERS.ini*. *FILELIST.list* should contain a list of files with either just the filenames to be processed or the full path to files to be processed. *PARAMETERS.ini* will control the processing to be performed, details of which are described in Sections 4 and 5.

4 Usage

4.1 Parameter Files

There are several example parameter files already included:

- AMBLOAD.INI - This will calculate the T_{sys} and gain (e.g. volts per Kelvin) from ambient load stare observations.

- DOWNSAMPLE.INI - This will downsample a data file in frequency by `factor` times and also check to see if any pointing needs to be added.

- FITJUPITER.INI - This will fix the pointing, downsample, and calibrate a Jupiter observation to the ambient load. Then it will fit a Gaussian to the time ordered data to derive amplitude, pointing and beam width measurements. It will also produce a calibration scale in units of Janskys/Kelvin for every horn and frequency channel.

5 Classes

5.1 BaseClass.H5Data

Useful functions for defining new classes:

- getdset - Retrieve a dataset, if it is not in memory load it.

- setdset - Load a dataset into memory.

- resizedset - Resize a dataset by passing it a new array.
- updatedset - Update a dataset values.
- getAttr - Get an attribute (stored in output file)
- setAttr - Set an attribute
- getextra - Get a dataset from the extra outputs
- setextra - Set an array to be assigned to extra outputs (must describe the shape of the array, e.g. which axis refers to horns, frequencies, etc...)
- resizeextra - Change dimensions of an extra dataset.

N.B. Never write directly to the dset or extras attributes of the H5Data class.

Useful attributes/functions for MPI routines:

- splitType - Axis type being split for MPI purposes (i.e., either Types._HORNS_, Types._SIDEBANDS_, Types._FREQUENCY_, Types._TIME_).
- selectType - Axis type being explicity selected (i.e., as above)
- selectIndex - Index being selected along selectType axis.
- splitFields - Names of fields in COMAP data structure that will contain a split axis.
- selectFields - Names of fields in COMAP data structure that will have a selected axis.
- hi/lo - Dictionary, for each splitField, defining where in the larger structure this process is accessing data.
- ndims - Dictionary containing dimensions of each dataset in memory for this process.
- fullFieldLengths - Dictionary containing dimensions of each dataset in memory in total.
- getDataRange - Function that returns how to split N values between M processes.