CS1 IDPPP

Algorithm

The IDPPP is the integration of three separate postprocessing steps to prepare LOFAR CS1 data for calibration and imaging.

- Divide out the bandpass using CS1_BandpassCorrector
- Flag the data using FrequencyFlagger, ComplexMedianFlagger, ComplexMedianFlagger2 or MADFlagger.
- Reduce the data in frequency resolution using CS1_DataSquasher

With the new IDPPP these can now all be executed with only one read and write of the data, and easy configuration of which steps to perform on the data.

Usage

The module is currently built as stand-alone ACC (Application and Control) executable. It depends on AIPS++/casacore so you will need to have initialised your environment to be able to use that. The application can be found in /app/LOFAR/stable on any of the storage or post-processing nodes of the CS1 off-line cluster. Using it is requires a two step approach. First you will need to write a parset (parameter set) file containing lines with settings on how you want to run the application. After you have written this parset file you need to execute the application in the same directory, where it will look for the parsetfile, by appending .parset to its own name, so make sure you named the file correctly.

You also need to have a logger properties file in your directory. Something like CS1_IDPPP.log_prop.

If the application finds the parset file, it will read the parameter setttings from it, and then execute the ACC *define*, *init* and *run* steps as defined in it's Process Control interface code. In practice this means that it will just “run”, as most of the ACC framework steps only have a real meaning in the context of the CEPFrame online streaming processing. The next section gives an overview of the parameters of the CS1_IDPPP

Parameters

The next parameters are available as input for the CS1_IDPPP currently available:

- **msin** - name of the measurmentset to use as input
- **msout** - name of the measurmentset to create
- **bandpass** - which bandpass to use [0=None, 1=CS1_BandpassCorrector]
- **flagger** - which flagger to use [0=None, 1=FrequencyFlagger, 2=ComplexMedianFlagger, 3=ComplexMedianFlagger2, 4=MADFlagger]
- **squasher** - which squasher to use, [0=None, 1=CS1_DataSquasher]
- **fixed** - selects which bandpass [1 to 4] to use, leave 0 if bandpass=0
- **freqwindow** Size of the frequency window. Leave at 1 except for MADFlagger
- **timewindow** Size of the time window. Leave at 1 except for both ComplexMedianFlaggers and MADFlagger
- **threshold** - the threshold above which to flag a datapoint. Used for FrequencyFlagger and MADFlagger.
- **algorithm** - To use RMS [0] or Median [1] for the FrequencyFlagger. Leave at 0 for

otherwise.

- **min** - Shortest baseline threshold in both ComplexMedianFlaggers. Leave at 0 for otherwise.
- **max** - Longest baseline threshold in both ComplexMedianFlaggers. Absolute Threshold for the MADFlagger. Leave at 0 for otherwise.
- **existing** - Do leave existing flags in the dataset if the algorithm doesn't flag the data. Used in all flaggers.
- **nchan** - Number of channels to squash.
- **start** - Start channel for squashing.
- **step** - step size for squashing.
- **skipflags** - Skip flagged data when using the DataSquasher. (only flagged data in the result if all data samples are bad)
- **allcolumns** - Process MODEL_DATA and CORRECTED_DATA if present.
- **timestep** - Number of timeslots to integrate

Example

```
fixed=5
freqwindow=5
timewindow=3
treshold=4.0
algorithm=0
min=1.0
max=1.0
existing=false
nchan=256
start=0
step=1
skipflags=true
allcolumns=false
msin=L2007_01055_SB0-1.MS
msout=test_out.MS
bandpass=1
flagger=4
squasher=0
```

Building

See the information on the Lofar build environment page for instructions how to build this software.

This is the basic command that should get the package built on lioffen:

```
autoconf_share/rub -confopt "--with-casacore=/app/casacore/casacore-0.3.0patched/ --with-wcs=/app/aips++/wcs_64"
```

Navigation

Back to Postprocessing Software.