SPAM frequently asked questions

This ever-expanding section is based on user feedback. For questions on installing SPAM, see this page. If you encounter a problem not listed below, please feel free to contact me.

SPAM pipeline failure

Q: Why does my SPAM pipeline run fail?

A: Although in general the SPAM pipeline is able to process many different data sets, there will be cases in which it fails. The output of the main pipeline (*process_target()*) is captured in a log file, located in the *datfil* directory. This will often provide guidance on the nature of the problem, but is a bit bulky. In your SPAM session, you can get a summary of the log file through

```
summarize_spam_log( './datfil/spam_<your_target_name>_*.log' )
```

Look for sudden increases in the noise, or decreases in the number of visibilities or the total cleaned flux. Something will have likely gone wrong between those two steps.

Too many open files

Q: SPAM crashes with a message about “too many open files”. What do I do?

A: Typical Linux installs have a limit on the number of files (or file descriptors) that a user can have open at any given time. The default is usually set to 1024 to prevent runaway programs from doing harm. As a user you can check that number: in (ba)sh, type

```
ulimit -n
```

or in (t)csh, type

```
limit descriptors
```

Often it is possible as a user to increase the limit on file descriptors. In (ba)sh, type

```
ulimit -n 4096
```

or in (t)csh, type

```
limit descriptors 4096
```

If this operation is not permitted, go talk to your sysadmin.

UPDATE: The latest ParselTongue distributed with SPAM has an improvement in place to reduce the

occurence of this error.

Out of pty devices

Q: SPAM crashes with a message about “pty devices”. What do I do?

A: Under certain circumstances, the SPAM pipeline allocates pty devices without freeing them after use. This was a flaw that has been fixed in ParselTongue. Installing ParselTongue version 2.3d or higher, distributed with SPAM, will resolve this issue.

Imaging in CASA

Q: Can I take the calibrated visibilities (.CAL.UVFITS) into CASA for imaging?

A: Yes you can. One limitation is that CASA from version 4.2 onwards doesn't allow stokes I visibilities to be imported. To overcome this, you can re-label the visibilities as being RR, and then image the RR visibilities in CASA. Converting the calibrated visibilities from stokes I to RR can be done in SPAM as follows:

```
uv = get_aips_file( 1, 'CALIBRATED', 'UVDATA', -1, 'UV' )
read_fits_uv( './fits/<target_visibilities>.CAL.UVFITS', uv )
convert_stokes_I_to_RR( uv )
write_fits_uv( uv, './fits/<target_visibilities>.RR.UVFITS' )
uv.zap()
```

Next, start CASA and run the *importgmrt()* task to convert the UVFITS data to a measurement set:

```
importgmrt( fitsfile = '<target_visibilities>.RR.UVFITS', vis =
'<target_visibilities>.RR.ms' )
```

Then run the *clean()* task with at least the following options:

```
clean( vis = '<target_visibilities>.RR.ms', imagename = '<target>', gridmode
= 'widefield',
    wprojplanes = <some number, e.g. 128>, stokes = 'RR', weighting =
'briggs', usescratch = True ... )
```

For more guidance on the options of CASA clean, go to the online CASA documentation.

Imaging with WSClean

Q: Can I image the calibrated visibilities (.CAL.UVFITS) with WSClean?

A: Yes you can. Similar to imaging in CASA, you need to convert the UVFITS data to a measurement set in stokes RR. Then, in the Linux shell, use command similar to the following (tested with WSClean v2.5):

```
wsclean -weight briggs 0 -pol RR -size 2048 2048 -scale 4.4asec -niter
100000 -auto-threshold 0.5 -auto-mask 4
    -multiscale -mgain 0.7 <target_visibilities>.ms
```

For more guidance on the options of WSClean, go to the WSClean wiki.

Problems running SPAM on new Intel CPUs, possibly after Linux update

Q: Why doesn't the SPAM pipeline work (anymore), with AIPS generating NaNs in certain tasks?

A: Updates of the Linux operating system in 2017 has triggered floating point problems when running AIPS tasks on a new line of Intel Xeon E5-xxx CPUs. This is likely caused by an outdated Intel compiler used to build the AIPS binary install. A GNU compiled version of AIPS 31DEC13 fixes the problem and is available here. It may run a bit slower than the Intel compiled version.

SPAM pipeline stops after finding no / too few peeled sources

Q: Is there a way to bypass the problem of SPAM finding no / too few peeled sources?

A: The SPAM pipeline needs to peel a sufficient number of sources to perform its direction-dependent calibration and modeling for ionospheric effects. There are several common cases in which SPAM cannot find enough peeled sources, mostly because of poor quality of observational data or because of dynamic-range limitations in the image. The user can force the pipeline to continue using the (direction-independent) self-calibration, but should be aware that the final pipeline results may be compromised. For this, the *allow_selfcal_skip* option needs to be enabled:

```
process_target( target_uvfits_file_name, allow_selfcal_skip = True )
```

SPAM crashes in screen session

Q: Why does my SPAM pipeline run crash when the screen session in which I run SPAM is interrupted?

A: As the SPAM pipeline can take several hours to days to run, it is useful to execute it in a screen session when working remotely. However, some users found that the SPAM pipeline crashed during a disconnect from the screen session with the error “Fatal IO error 11 (Resource temporarily unavailable) on X server localhost:1.0.”, or something similar. This is likely due to the fact that the screen is started in an environment with the X server available that becomes unavailable after the disconnection, causing the crash of SPAM. A solution might be to use a “fake” X server, such as xvfb, in the screen session. Another solution is to run SPAM remotely via a VNC session. Note that checking

the progress of the pipeline can also be done by checking the spam*.log in the datfil subdirectory.

(Kindly contributed by Andrea Botteon)

Manual flagging

Q: Can I manually flag bad data in SPAM?

A: Manual flags can be inserted in both the *pre_calibrate_targets()* and the *process_target()* steps. Both functions allow for a list of python dictionaries to be passed as the argument *manual_flags*. The dictionaries consist of keys and values that correspond to the adverbs and values of the AIPS task *UVFLG*. Here is an example:

```
manual_flags = [ { 'antennas' : [ 5, 12 ] },
                 { 'antennas' : [ 1 ], 'stokes' : 'RR' },
                 { 'antennas' : [ 23 ], 'timerang' : [ 1,23,45,0, 99,0,0,0 ]
} ]
pre_calibrate_targets( uvfits_file_name, manual_flags = manual_flags )
```

In this example, the antennas with AIPS indices 5 and 12 (corresponding to CASA indices 4 and 11) will be flagged completely (all times, both polarizations), for antenna 1 only stokes RR will be flagged (all times), and antenna 23 will be flagged from timestamp 1,23,45,0 [=day,hour,min,sec] onwards (both polarizations).

(Kindly suggested by Chris Riseley)
