Testing of NDPPP

David Rafferty

1. Overview

Limited testing of NDPPP has been done, concentrating mainly on the parameters relevant for RFI flagging. Although further testing would be useful, NDPPP was found to be a significant improvement over IDPPP in both speed and functionality. A comparison was also made between NDPPP and rficonsole, a flagger available outside of NDPPP, in flagging performance and speed. Although NDPPP was found to be approximately 2.5 times faster than rficonsole when optimum use of a compute node is made, the flagging performance of rficonsole was found to be considerably better. It is therefore recommended that rficonsole be used for RFI flagging in the standard imaging pipeline unless time considerations are absolutely critical.

If RFI flagging is performed with NDPPP, the parameters listed in the following section are recommended to be used for NDPPP in standard runs of the imaging pipeline. These parameters should result in both improved RFI flagging and a factor of ≈ 2 decrease in run time compared to IDPPP runs in the current pipeline. Sections 2 and 3 present the results of averaging and flagging tests and give comparisons to IDPPP and rficonsole.

1.1. Suggested Parameters for NDPPP

The following parset is suggested for future runs of the imaging pipeline. This NDPPP run can be used as a direct replacement for the current IDPPP run, as it performs the flagging and squashing functions currently done in the pipeline reduction. With these parameters, testing indicates that most of the moderate-to-strong RFI (i.e. that generally visible to the eye) is identified and flagged and that NDPPP runs on a single subband on a single processor in $\approx 1/4$ of the observation duration (i.e., 30 minutes run time for a 2-hour observation) for a 20-station dataset. It is expected that the run time scales $\sim$ linearly with the number of baselines, although this was not tested.

When using the parameters given in this section, NDPPP was found to flag most of the visible RFI. However, due to the low threshold used in the initial flagging passes, a significant quantity of good data (judged to be $\sim 10 - 15\%$) is also flagged. A variety of thresholds were tested, and higher thresholds generally resulted in less good data being flagged but also more strong RFI being missed. Further tests should be made in order to determine the ideal threshold, which may differ for LBA and HBA data.

```
msin = <input_ms>
msin.startchan = 8
msin.nchan = 240
msin.datacolumn = DATA
msout = <output_ms>
msout.datacolumn = DATA

steps = [flag1,flag2,avg1,flag3]

# Squashing pass to average all channels into one
```

```
avg1.type = squash
avg1.freqstep = 240
avg1.timestep = 1

# Flagging pass 1 (on unsquashed data with medium time window, XX & YY only)
flag1.type=madflagger
flag1.threshold=2
flag1.freqwindow=101
flag1.timewindow=1
flag1.correlations=[0,3]

# Flagging pass 2 (on unsquashed data with medium freq window, XX & YY only)
flag2.type=madflagger
flag2.threshold=2
flag2.freqwindow=1
flag2.timewindow=101
flag2.correlations=[0,3]

# Flagging pass 3 (on squashed data with wide time window, all corr)
flag3.type=madflagger
flag3.threshold=3
flag3.freqwindow=1
flag3.timewindow=901
flag3.correlations=[0,1,2,3]
```

2. Averaging Tests

A test of averaging in frequency was performed using SB98 of the L2010_05703 observation (a 6-hour LBA observation of a point source field with 11 stations) using one processor (on lce022). Averaging was done in frequency with the "avg1" parameters given in Section 1.1; no flagging was done. NDPPP completed the averaging in 3m25s versus 11m23s for IDPPP.

3. Flagging Tests

A number of flagging tests were performed using an unsquashed 2-hour subset of SB50 of the L2009_16167 observation (a 60-hour HBA observation of 3C61.1 with 20 stations), again using one processor (lce022 or lce023). Testing was done with the MADFlagger, which is by far the best of the IDPPP flaggers (and the only one available in NDPPP at this time). Details of the test are given in the following sections.

3.1. Flagging Using a Subset of Correlations

NDPPP allows flagging using any combination of correlations. An extensive survey of RFI in existing LOFAR data shows that XX and YY often have very different RFI at a given time. Therefore, flagging using only XX or only YY does not produce good results. However, RFI present in the XY and YX correlations is usually also present in XX and YY. Therefore, in the limited testing done to this point, flagging on both XX and YY appears to be sufficient to identify most of the strong RFI. In the strategy given in Section 1.1, the initial flagging runs (performed on uncompressed data) are done on XX and YY, with a further flagging run done after compression using all four corellations. This strategy was found to be a reasonable compromise between flagging effectiveness and processing time. Lastly, since NDPPP ignores data that has already been flagged in another correlation, the order in which correlations are specified can affect the run time. Therefore, when possible, correlations should be specified in the parset in decreasing order of RFI contamination (i.e., the correlation with the worst RFI should be specified first).

3.2. Multiple Flagging Passes in a Single NDPPP Run

NDPPP allows for multiple flagging (and squashing) passes in a single run. Runs were successfully made with one, two, and three flagging passes. Successful runs were also done that included multiple flagging passes and one averaging pass.

3.3. Comparison to IDPPP

A number of test runs were made with both NDPPP and IDPPP using the unsquashed 2-hour 20-station dataset described above. The times required are summarized in Table 1 (the window used in each MADFlagger run is given by TW, the time window in number of time slots of 3 seconds each, and FW, the frequency window in number of channels; a threshold of 2 was used for all runs). NDPPP requires $\approx 35\%$ less time than IDPPP when all correlations are used for flagging, and is up to ~ 4 times faster when NDPPP uses just one or two correlations. Note, however, when just one correlation is used, flagging performance is poor. When XX and YY are used, the flagging performance is judged to be essentially as good as that obtained using all four correlations.

3.4. Comparison to rficonsole

The rficonsole flagger is available outside of NDPPP and uses a number of sophisticated techniques to identify RFI that are not used in the NDPPP flagger. A description of these techniques is beyond the scope of this report; see Offringa et al. (2010) for a detailed description of rficonsole. When run

Table 1. Comparison of NDPPP and IDPPP.

Run	NDPPP	IDPPP
FW=31, TW=5	32m1s (XX,YX,XY,YY)	51m12s (XX,YX,XY,YY)
FW=101, TW=1, then FW=1, TW=101	27m42s (XX,YY)	2h1m (XX,YX,XY,YY)
FW=1, TW=21	5m58s (XX)	21m50s (XX,YX,XY,YY)

with the default flagging strategy,[1] rficonsole uses all four correlations for flagging and is very effective. A comparison between NDPPP and rficonsole in flagging effectiveness and speed was done, using a 6-hour, 22-station HBA observation of Virgo A (L2010_20313). The first stage of the NDPPP strategy described in Section 1.1, which flags on only the XX and YY correlations, was used for the purposes of this comparison. The latest version (as of September 2010) of rficonsole was used.

To make the fairest comparison possible, the best average time per subband achievable by the flagger was calculated. For each flagger, a number of runs with a differing number of simultaneous instances were made, up to 8 subbands (each compute node has 8 processors; therefore 8 subbands will maximally load a single node). The average time per subband for each run was then calculated. Table 2 summarizes the results.

The best average time per subband for the L2010_20313 dataset is 12.5 minutes for NDPPP and 29 minutes for rficonsole. Runs with rficonsole of more than one subband require longer times per subband, probably due to i/o saturation. Therefore, rficonsole runs most quickly with one subband per node and at best will require approximately 2.5 times longer to run through a typical dataset than NDPPP. However, due to its use of more sophisticated flagging techniques, rficonsole flags visible RFI more effectively while avoiding a large number of false positives (rficonsole typically flags $\lesssim 1\%$ of good data, compared to $\sim 10 - 15\%$ for NDPPP run with the settings given in Section 1.1; see Figure 1 for a comparison of the flagging results obtained with rficonsole and NDPPP for data with strong RFI). Therefore, if the longer runtime is acceptable, rficonsole is clearly the preferred flagger.

4. Functionality Not Yet Tested

A number of features of NDPPP were not tested. These are: the PreFlagger, selection of subsets of the data by time, flagging on columns other than DATA (e.g., CORRECTED_DATA), and updating/preserving flags in an existing MS, among others.

REFERENCES

Offringa, A. R., de Bruyn, A. G., Biehl, M., Zaroubi, S., Bernardi, G., & Pandey, V. N. 2010, MNRAS, 405, 155

[1]The strategy may be altered with the “rfistrategy” program.

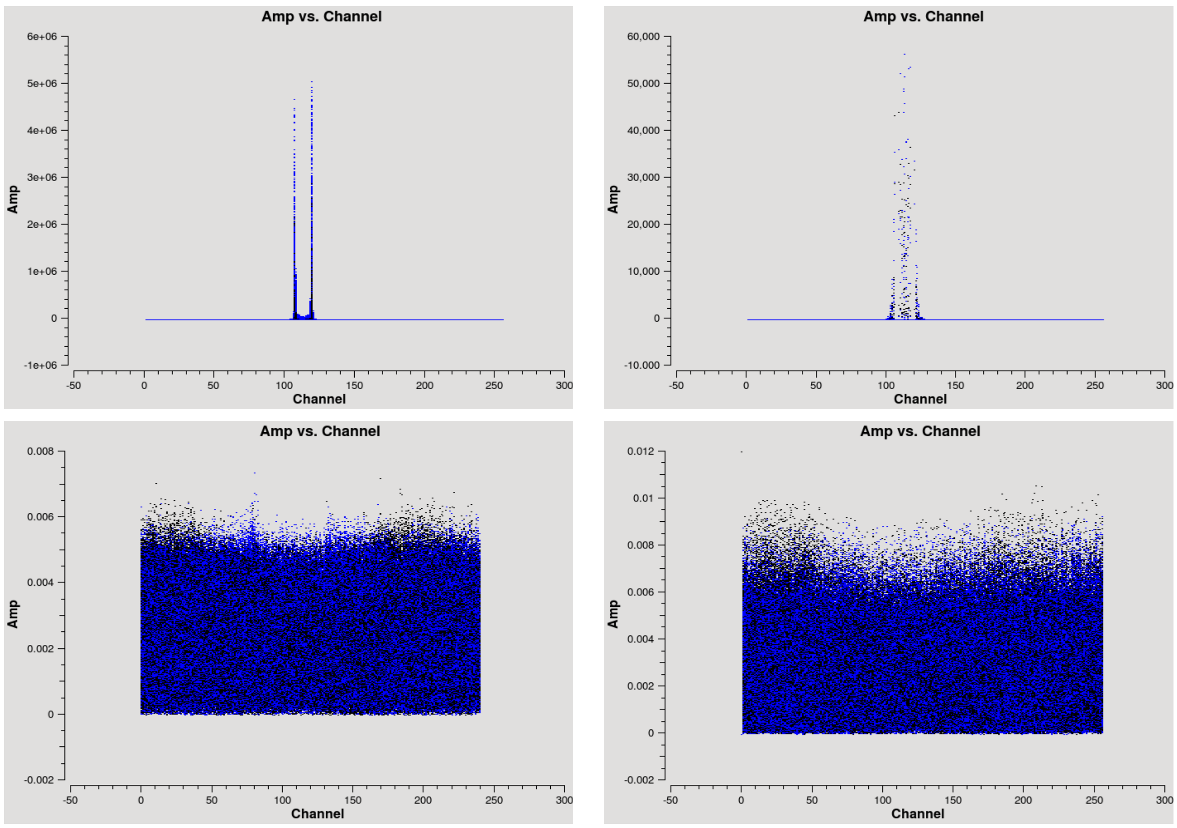

Fig. 1.— Comparison of flagging performance for XX (black points) and YY (blue points) correlations for NDPPP and rficonsole for SB50 of the L2009_16167 dataset. The top-left plot shows amplitude vs. channel for unflagged data. The top-right plot shows the results of flagging using NDPPP with the current pipeline parameters (FW = 31 channels, TW = 5 time slots). The bottom-left plot shows the results of flagging using NDPPP with the suggested parameters given in Section 1.1. The bottom-right plot shows the results of flagging using rficonsole with the default parameters.

Table 2. Comparison of NDPPP and rficonsole.

Run	Run time	Average time per SB
NDPPP		
1 SB, 1 instance	1h15m	1h15m
2 SBs, 2 instances	1h15m	37.5m
4 SBs, 4 instances	1h15m	18.75m
8 SBs, 8 instances	1h40m	12.5m
rficonsole		
1 SB, 1 instance	2h10m	2h10m
1 SB, 7 instances	29m	29m
1 SB, 8 instances	32m	32m
2 SBs, 8 instances	1h12m	36m
4 SBs, 8 instances	6h4m	1h16m