Cobalt commissioning report

M.A. Brentjens

September 1, 2014

Contents

Chapter 1

Introduction

Cobalt is the correlator that was supposed to succeed the BG/P at 2013-12-31 at the latest (Broekema et al. 2013). Due to some delays, it has actually replaced the BG/P at the end of April 2014. In the scenario-1 design (Broekema 2013), Cobalt comprises 8 PC nodes, equipped with 2 NVIDIA GPU units each, interconnected by an infiniband switch.

This document describes in as much detail as possible, from a technical-scientific point of view, *what* must be tested, *how* it should be tested, and when and how it was tested. At the end of commissioning, we want the correlator to

be feature complete: it must be able to perform the same types of observations as the BG/P;

be correct: the Cobalt output must be the same as the BG/P output, within the expected numerical precision;

perform well: support all BG/P observation types with an execution speed the same or better than the BG/P;

be robust: the correlator should not crash on errors in the specification, and if it does, it should leave clear error messages.

well integrated: we must be able to simply specify and observation in MoM, schedule it, have it run on Cobalt, get data written to CEP-2, pipelines started, and obtain pipeline products that can be ingested into, and retrieved from the long term archive (LTA).

Because Cobalt is built from commodity hardware, and all interesting functionality is implemented in software (Mol 2013) running on CPUs and GPUs, many experiments described in this document can be conducted with synthetic or pre-recorded data.

In this document, the word "pipeline" refers to specific Cobalt code that processes incoming data during an observation, not post processing software that usually runs after an observation has completed, unless explicitly stated otherwise.

Chapter 2

Basic sanity

Exp. 1 [PASSED 2013-11-21]: Reproducibility (correlator) [recorded data]
Aim: *Running the same input UDP streams through cobalt using the same parset must produce bit-for-bit the same output data.*
This is a very simple experiment. Just play back the same observation twice and compare the output. After some initial issues with the circular buffer code, this test passed on data set L189429. We used the following function for validation.

```
def are_they_identical(ms_file_name, ms_ref_file_name):
    tab     = table(ms_file_name).query('ANTENNA1 != ANTENNA2')
    tab_ref = table(ms_ref_file_name).query('ANTENNA1 != ANTENNA2')
    data     = tab.getcol('DATA')
    data_ref = tab_ref.getcol('DATA')
    uvw     = tab.getcol('UVW')
    uvw_ref = tab_ref.getcol('UVW')
    if len(uvw) != len(uvw_ref) or abs((uvw - uvw_ref)).max() != 0.0:
        raise ValueError('Uvw coordinates differ.')
    uv_distance = sqrt(uvw[:,0]**2 + uvw[:,1]**2)
    diff        = data - data_ref
    if abs(diff).max() != 0.0:
        raise ValueError('Data sets are not identical')
    return True
```

Exp. 2 [PASSED 2014-03-10]: Reproducibility (beam former) [recorded data]
Aim: *Running the same input UDP streams through cobalt using the same parset must produce bit-for-bit the same output data.*
Repeat the previous experiment for the beam formed modes. Do this for both coherent stokes, incoherent stokes, with, and without coherent dedispersion.

This test has been incorporated in the automated test suite, and runs daily. Coherent dedispersion is not yet included.

Chapter 3

Channels

The channelization of the input sub bands is the first stage in both the correlator pipeline and the beam former pipeline.

Exp. 3 [PASSED 2013-06-05]: Channel isolation [synthetic data]
Aim: *Measure the band pass of individual correlator output channels.*
Generate synthetic CW input signals at a prime number of frequencies, beginning at the centre of one channel, and increasing the frequency until at the centre of a neighbouring channel. Use e.g. 44 frequencies, where the first and last should be precisely at channel centres. This leaves 43 intervals and avoids resonances with other constants in the code base. Plot the real part of the output as a function of frequency.

This experiment has been conducted on 2013-06-05 by Wouter Kleijn. The bandpass is plotted in Fig.3.1.

Exp. 4 [PASSED 2013-11-21]: Frequency labelling: coarse [recorded data]
Aim: *Determine the correctness of frequency labelling in the output MS.*
There are two strong transmitters in the HBA_LOW band at sub band 357 that can help verify the labelling of channel frequencies, as well as the order in which channels are written. A channel's frequency label represents the channel's *centre.* The transmitters to look for are

- P2000 at 169.650 MHz;
- KPN ERMES at 169.750 MHz.

The central frequency (middle of channel 128, counting from 0) of sub band 357 is 169 726 562.5 Hz. When operating at 256 channels per sub band, the channel width is 762.939453 Hz. The P2000 peak should therefore show up in channel 28 (counting from 0). In fact, the peak must be 0.148 channel widths, or 113 Hz, from the lower boundary of channel 28. KPN ERMES, if within reach, should peak in channel 159, about 0.22 channel widths, or 168 Hz, from the lower boundary of the channel, but the exact position within a channel is not easy to extract from this test because the carriers are so narrow.

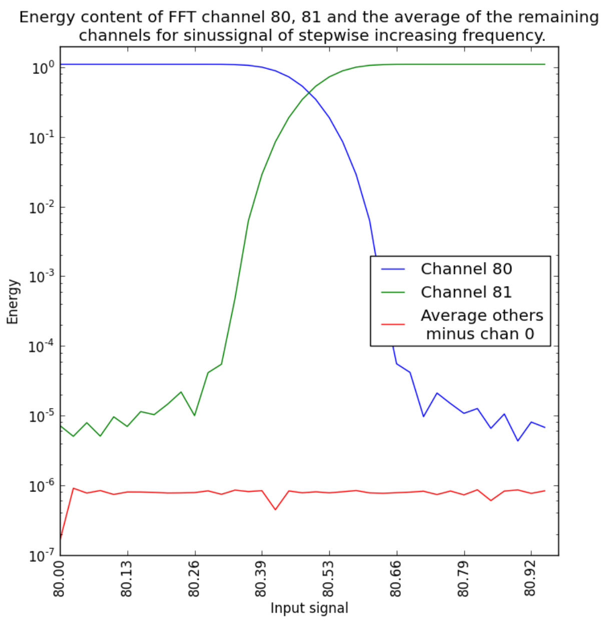

Figure 3.1: Measurement of channel band pass and channel separation by injecting a sinusoidal signal at various frequencies.

For this experiment, pre-recorded station data of this sub band should be fed to Cobalt. The auto-correlated visibilities must be averaged to 1 second, and the MS stored. Only 10 seconds of data from one station are required.

Considering that the beam forming pipeline and the cross correlating pipelines have different approaches to split a sub band into channels, this should be conducted for *both* pipelines. Ideally, this experiment is conducted for 200 MHz as well 160 MHz data.

This sub band ends up being SB018 in the test data sets. This was tested with data set L189429 (16 bit, 3C 295) on 2013-11-20. The associated plot is shown in Fig. 3.2.

We did not yet conduct the 160 MHz experiment.

Exp. 5 [PASSED 2014-01-21]: Frequency labelling: fine [synthetic data]

Aim: *Determine correctness of frequency labelling within a channel*

Prepare a timeseries by taking a spectrum with a gaussian amplitude as a function of frequency, where the peak of the gaussian is about 1/3 of a channel away from the channel edge with a σ of about 2 channels, and Fourier transforming this spectrum to a time series at a time interval of 5.12 μs.

Feed the time series to Cobalt, and look at the generated sub band. Fit a gaussian to the channel amplitudes and determine its peak. It should correspond to the exact frequency that was originally put in.

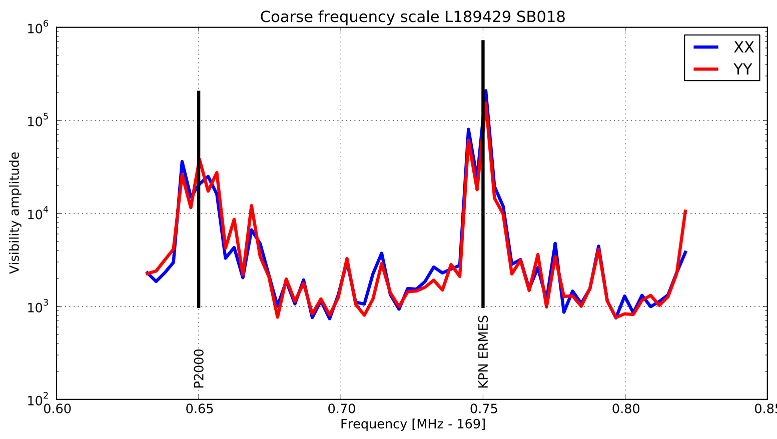

Figure 3.2: Spectrum of sub band 357 containing the ERMES and P2000 signals for data set L189249. This is the average spectrum over all baselines, both cross correlations and auto correlations. The vertical black lines are at the listed carrier frequencies for these transmitters.

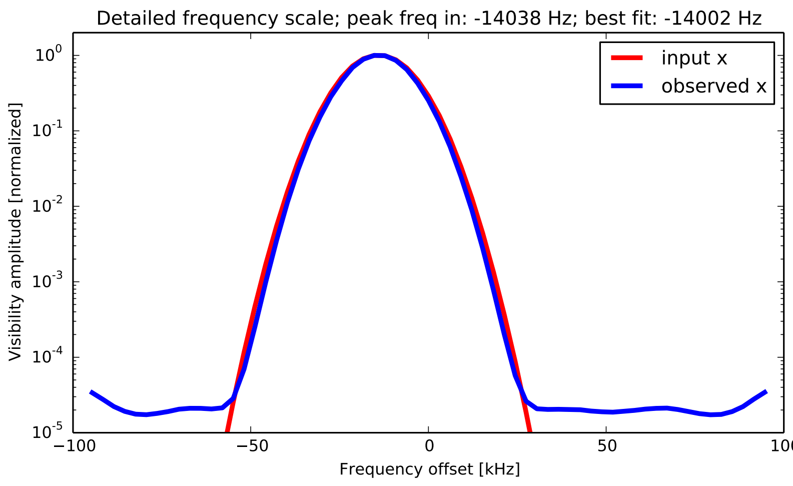

Figure 3.3: Spectral response of the 64 bit float precision time series (red) and the autocorrelation spectrum of the corresponding 8-bit integer time series, as processed by Cobalt without correcting for the station sub band bandpass. Channel width is about 3 kHz, so frequency labelling is approximately good to 1% of the channel width.

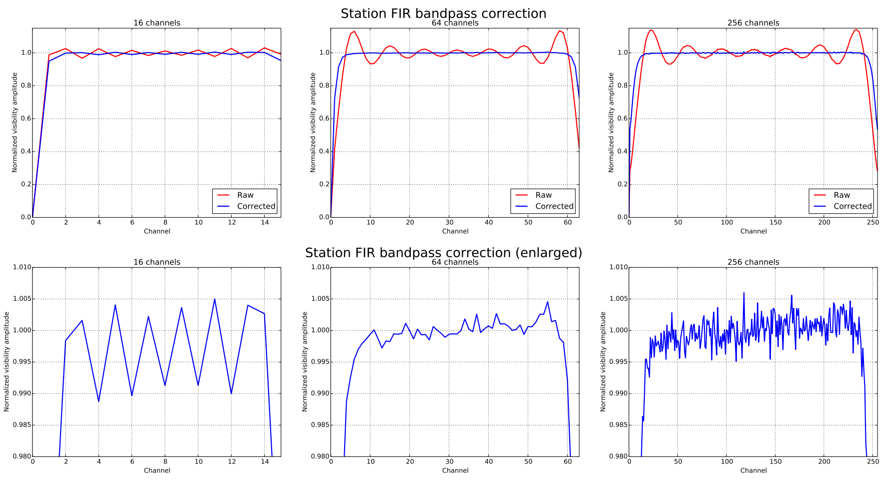

Figure 3.4: Station sub band band pass correction on / off (correlator). Note the scale in the bottom panel.

We have done this experiment slightly differently, generating the time series from a complex Gaussian noise time series, FIR filtered with a filter response in the frequency domain that had a 12 kHz standard deviation. The exact procedure is described in ipython notebook frequency-scale-cobalt.ipynb, which is available from M.A. Brentjens upon request.

The results are shown in Fig. 3.3. We have verified that the labelled central frequency of the channels is good to about 1% of the channel width. This is good enough for all science cases we are aware of. The current experiment gives no reason to believe that the frequency labelling is less than perfect.

Exp. 6 [PASSED (corr) 2013-11-22 / INSUFFICIENT (beamformer) 2014-08-28]: Station band pass correction [recorded data]
Aim: *Verify that station band pass correction works.*
The stations use a poly phase filter (PPF) to separate the antenna's time series into 512 sub bands. This leaves a distinctive wobbly pattern in a sub band's band pass. Cobalt must be able to correct this.

Pre-record 24 adjacent HBA sub bands in an area of spectrum that is relatively free of interference. 10 seconds of data would suffice. Use two core stations and two remote stations. Produce plots of the amplitude and phase of the concatenated ten sub band spectrum for all combinations of settings in the following grid:

Correction	16 ch/sb	64 ch/sb	256 ch/sb
on (cross-corr)	DONE	DONE	DONE
off (cross-corr)	DONE	DONE	DONE
on (beam formed)	DONE	DONE	DONE
off (beam formed)	DONE	DONE	DONE

In reality, we have averaged all baselines, and in the beam former case, all

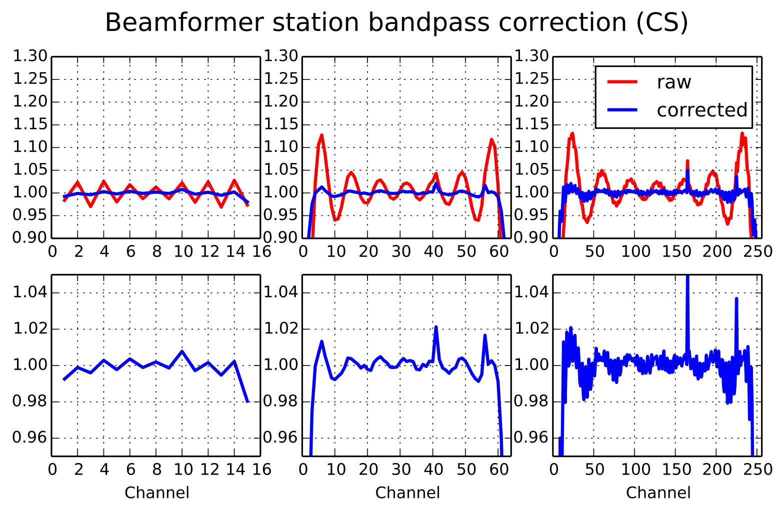

Figure 3.5: Station sub band band pass correction on / off (beamformer coherent stokes). Note the scale in the bottom panel.

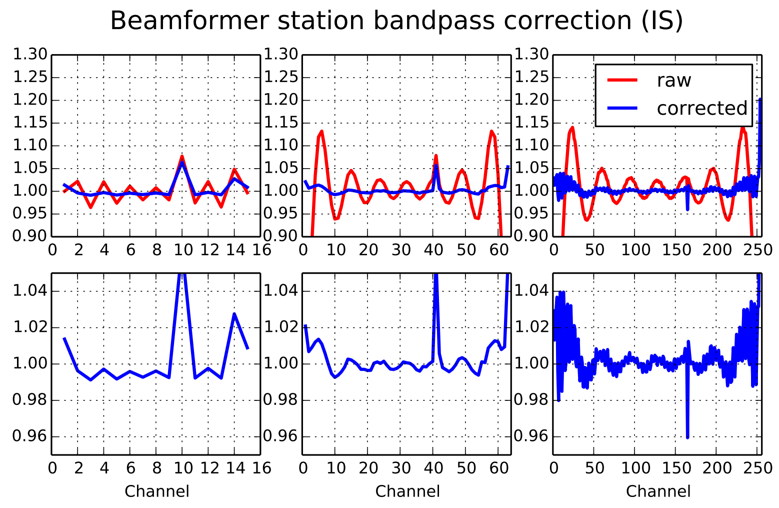

Figure 3.6: Station sub band band pass correction on / off (beamformer incoherent stokes). Note the scale in the bottom panel.

core stations. Plotting the phase was therefore deemed not very useful, and meaningless in the beam former case, because the output spectra there are real valued anyway.

We used sub bands 113. . . 136 (122.07 –126.56 MHz central frequencies), which are generally very clean. in terms of RFI. In fact, we used all Dutch stations except for CS013, CS401, and RS210 in HBA_DUAL_INNER mode. The observation lasted for 16 seconds. The first and last three seconds were discarded to avoid potential start/end effects. We observed Cyg A near transit. Figure 3.4 shows the results for the correlator. It is clear that the correction works the same way it did on the BG/P. The residual slope is due to the mean antenna and receiver band pass in the lower part of the HBA band.

For the beamformer (Figs. 3.5 and 3.6) the correction is reasonable, but there is some residual from the station band pass. Additionally, there appears to be small ripple as a function of frequency. These things needs to be investigated further during the regular operational phase for Cobalt. Given that the residual shape is bigger in the IS case, I suspect that it has something to do with the station's autocorrelations, which are not included in the correlator plot. They are included (along with all cross correlations) in the CS plot, and exclusively make up the IS plot.

Chapter 4

Delay compensation

Exp. 7 [PASSED 2013-11-15]: Order of antennas and sign of baseline vector [recorded data]
Aim: *Verify that ANTENNA1≤ANTENNA2 and that UVW = ANTENNA2 - ANTENNA1*
Run the test case `test_ant_columns` from the program `verify-ms-format.py` listed in appendix A. This was done for L188487, which passed the test case on 2013-11-15.

Exp. 8 [PASSED 2013-11-20]: Single precision versus double precision [recorded data]
Aim: *Can the sin and cos calculations in fringe stopping be done in single precision?*
Assess the difference between doing fringe stopping in single precision versus double precision. Use the double precision as a proxy for the truth. The main aim is to determine the effect of rounding error in the sin and cos calculation in the fringe stopping routines on the visibilities that are produced.

Our conclusion, based on experiments with data set L189429, is that the RMS fractional error on the visibilities at 1 s and 3.052 kHz time- and frequency resolution is about $3 \cdot 10^{-5}$. As can be seen in the right panel of Fig. 4.1, the histogram of deviations in the ratio of double / single precision data is nicely symmetrical. Given sufficiently large integration times and frequency ranges, these rounding errors will behave stochastically, leading to an increase of the noise level of the visibilities.

The noise contribution due to this problem at the given time- and frequency resolution is exactly the visibility amplitude multiplied by $3 \cdot 10^{-5}$. In case of noise-limited data, this leads to an imperceptibly small increase of the total noise, because the thermal noise and numerical noise add in quadrature. For the rounding noise to become dominant, the total field flux has to be about 1.4 MJy in the HBA, and much more in the LBA. Keep in mind that this just raises the noise level, but that the noise still averages down with the square root of the time-frequency window.

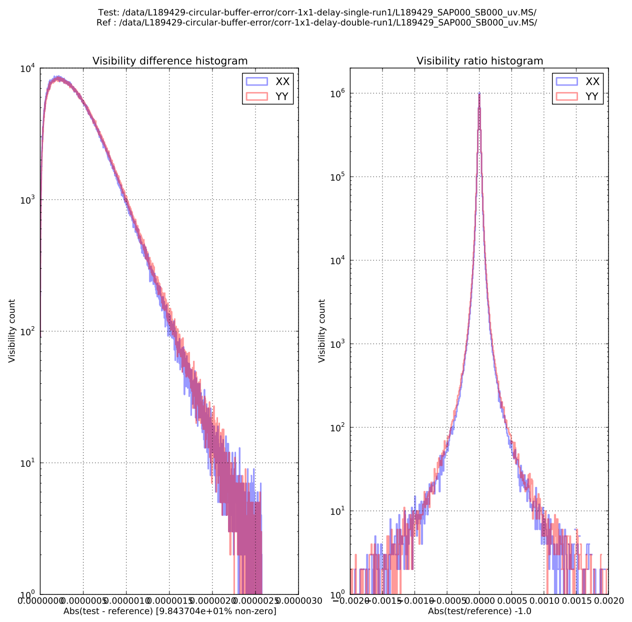

Figure 4.1: Effect of single precision in sin cos calculations for fringe stopping. Histogram of differences and ratios.

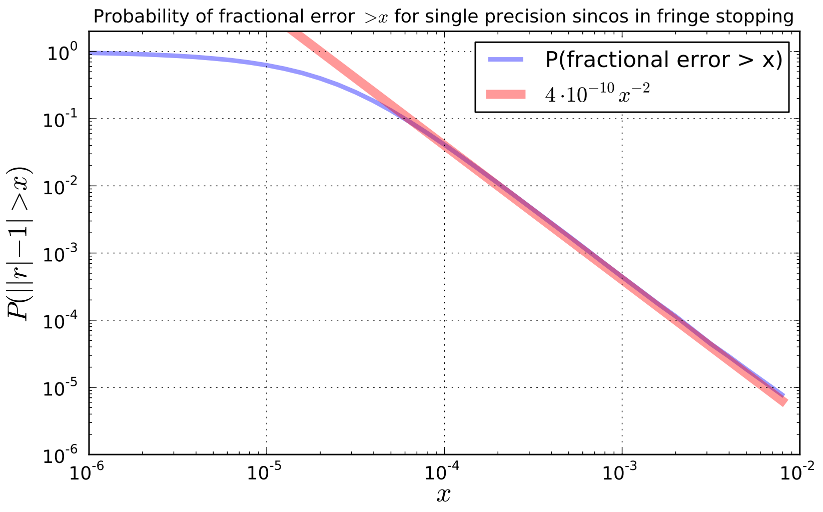

Figure 4.2: Effect of single precision in sin cos calculations for fringe stopping. Cumulative PDF.

Figure 4.2 shows the cumulative probability density function. That is, the probability that a visibility at the 1 s – 3 kHz resolution has a fractional error *larger* than x.

Exp. 9 [PASSED 2013-12-19]: Setting of static phase/delays per pol and antenna field [recorded data]
Aim: *Verify that phases and delays are applied per polarization and antenna field*
Use prerecorded data of 3C 196 in the HBA, and Cyg A in the LBA. Two minutes of data should suffice. The experiment consists of the following steps. For both data sets:

1. Correlate the data, applying the current delay values in use for the BG/P.: PASSED 2013-11-01

2. Derive the delays with respect to the mean delay for all core stations and verify that it is less than 1 ns for all stations. If this is the case, the delays are applied with the correct sign.

3. For one station, and 100 ns to the delay of the Y polarization only.

4. For another station, add 70 degrees of phase to the X polarization only.

5. Re-correlate the data again, and fit for delays and phase at 0 MHz frequency, and verify that the 100 ns and 70° changes show up in the appropriate stations and polarizations. PASSED 2013-12-19

Exp. 10 [PASSED 2013-11-15]: Station UVW coordinates [recorded data]
Aim: *Verify the J2000 UVW coordinates in the MS*
For core–core, remote–remote, and ILT–ILT baselines, verify that the J2000 UVW coordinates in the measurement set are equal (to within 5 cm) to the UVW coordinates derived from the station's ITRF positions, transformed using casacore for the epoch of the observation.

Run the test case `test_uvw_casacore` from the program `verify-ms-format.py` listed in appendix A. This was done for L188487, which passed the test case on 2013-11-15.

Exp. 11 [PASSED 2014-01-31]: Baseline phases [10 h observation on 3C 48 or 3C 147]
Aim: *Ensure that there are no sudden jumps in the visibility phases or delays*
To test if the delays are compensated in a smooth way, and no major interpolation accidents occur, make plots of the baseline delays and phase offsets for all

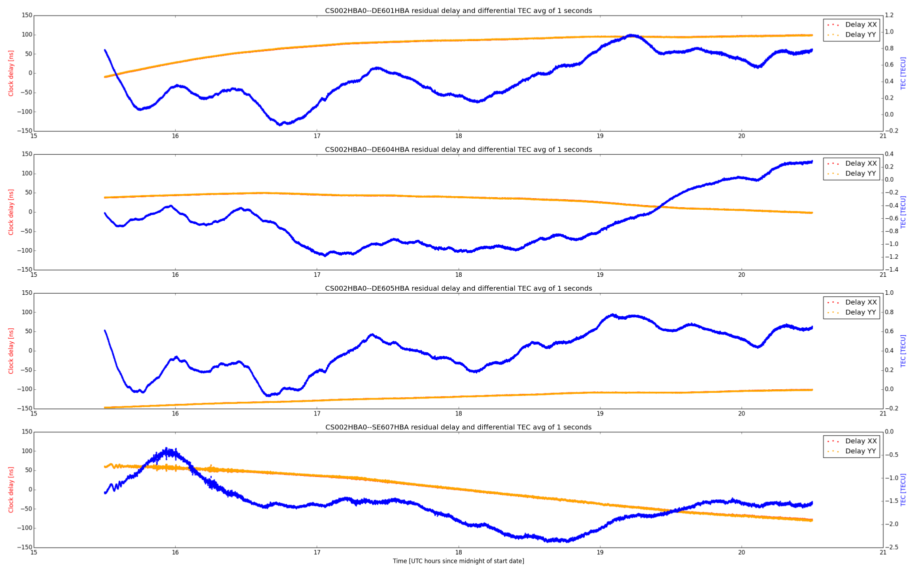

Figure 4.3: Clock delay and TEC difference between CS002HBA0 and four international stations during observation L203395.

international and remote baselines from a long HBA observation of a compact source.

All delay- and phase structure should be attributable to station clocks, the ionosphere, and slow moving differences between the casacore and CALC delay models.

An observation of 3C 48 was conducted on January 31st (L203395). This observation involved all Dutch and international stations simultaneously in HBA_DUAL mode and lasted for about 5 hours. We fitted for phase 0 offsets and delays for the XX and YY correlations of between CS002HBA0 and the international stations. DE601, DE604, DE605, and SE607 provided good data.

Figure 4.3 shows the clock and TEC differences. Uncertainties for the delays are typically a bit less than a ns. TEC uncertainties are typically a few mTECU. It is clear that there are fairly large systematic delay variations of up to 150 ns. The BG/P showed similar behaviour. This is most likely due to the way we invoke the casacore delay model or problems in the actual casacore delay model. Because decorrelation due to this problem is rather benign at 4 channels per sub band resolution, we intend to solve this after April 1st.

Figure 4.4 shows the delays again, but after subtracting a 360 s running mean. This figure will highlight any sudden delay jumps, if any. Based on this figure, we can confidently state that there are no sudden jumps in delay larger than about half a ns on timescales smaller than 6 minutes. The structure in the CS002HBA0–SE607HBA baseline is due to the central source being partially re-

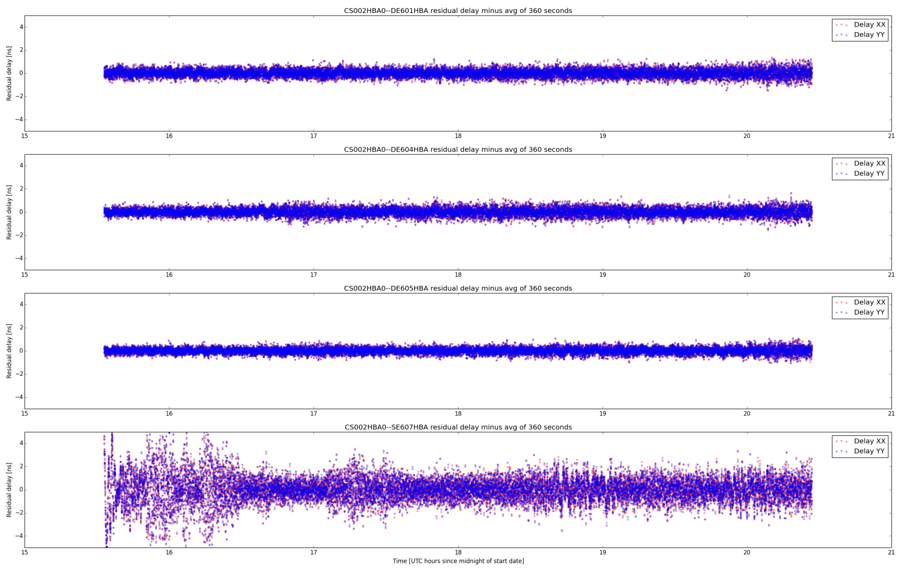

Figure 4.4: Clock delay difference between CS002HBA0 and four international stations during observation L203395 after subtracting a 360 s running mean.

solved out, and other sources beginning to contribute to the visibility in a significant way.

Chapter 5

Correlator

Exp. 12 [TODO]: MS *uvw* coordinates versus delay compensation *uvw*s [recorded data]
Aim: *Verify that the uvw coordinates in the MS correspond to* $\langle \vec{u}_j - \vec{u}_i \rangle$
The *uvw* coordinates in the MS for a given visibility must be equal to the mean of the difference between the *uvw* coordinates for the stations in J2000, averaged over the integration time of the visibility. Testing this requires an experiment in which the correlator logs the offset between a station and CS002LBA in the J2000 frame for at least 2 stations, at *every* 5 μs time step for one full integration. Obviously, this is an experiment that may not be possible to run in real time, but it needs to be done to ensure that at the very least the delay model and the MS are consistent.

A way to store correlator-computed station UVWs in the MS is currently [2014-03-11] being worked on. It has not yet been installed.

Exp. 13 [PASSED? 2013-11-22]: Flux scale 16-bit versus 8-bit mode [recorded data]
Aim: *Verify that 16-bit and 8-bit data have the same flux scale*
At the stations, 8-bit data are normalized differently than 16-bit data, leading to a voltage amplitude that is a factor 16 lower for 8-bit data. The BG/P does not correct for this difference. In Cobalt, we do want to correct this to ensure a uniform flux density scale, independent of bit mode.

The experiment is very simple: correlate 8-bit data and 16-bit data, recorded within minutes of each other, straddling transit of 3C 196, and compare the visibility amplitudes. They must be the same to within a couple percent. In the BG/P, they would be different by a factor 256.

This experiment has not yet been done properly, but a preliminary test with L189429 (16 bit) and L189430 (8 bit) observations of 3C 295 was done, while 3C 295 was a fair distance away from transit. We ran the following code on locus004:

```
from pyrap.tables import table
ms16 = '/data/L189429-reference-single/L189429_SAP000_SB000_uv.MS/'
ms08 = '/data/L189430-reference-single/L189430_SAP000_SB000_uv.MS/'
```

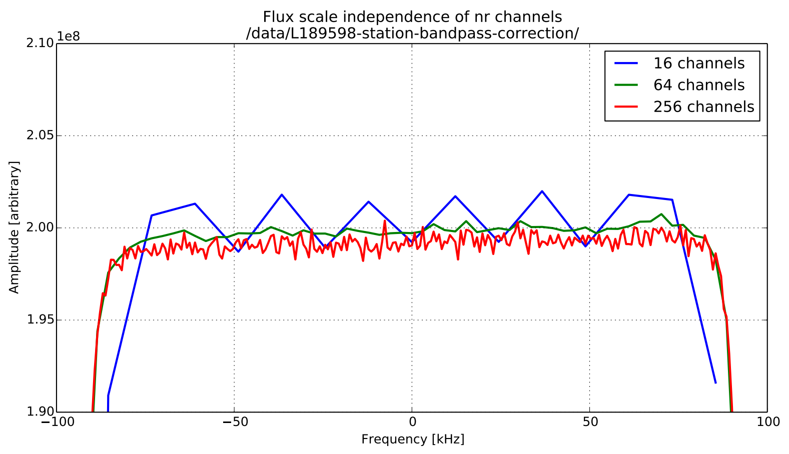

Figure 5.1: Flux scale is independent of the number of channels.

```
sel16   = table(ms16).query('SQRT(SUMSQUARE(UVW[1:2])) > 1000.0')
sel08   = table(ms08).query('SQRT(SUMSQUARE(UVW[1:2])) > 1000.0')
data16  = sel16.getcol('DATA')
data08  = sel08.getcol('DATA')
median16 = median(abs(data16)[:,6:-6,0::3]) # avg xx and yy
median08 = median(abs(data08)[:,6:-6,0::3]) # avg xx and yy
print '8 bit: %.3f; 16 bit: %.3f' % (median08, median16)
```

Which gave "8 bit: 0.027; 16 bit: 0.020" as an answer. These numbers are within 25% of each other, and certainly not a factor of 256 off. We are therefore fairly confident that this actually works.

Exp. 14 [PASSED 2014-01-29]: Flux scale independent of nr channels [recorded data L189598 Cyg A]
Aim: *Ensure that the mean amplitude in each channel is the same, independent of the number of channels specified.*
In the BG/P, Fourier transforms from the time to the frequency domain were not normalized, leading to a flux scale that is proportional to the number of output channels. This is clearly not desirable. For Cobalt, the time to frequency transforms should be normalized, ensuring a uniform flux scale.

This can be tested with the same data sets that were used for the station sub band band pass correction test. Fig. 5.1 shows the results for 16, 64, and 256 channels.

Exp. 15 [CANCELLED]: Equivalence to BG/P in 16 bit mode [recorded data: 3C 295 all stations]
Aim: *Assert that Cobalt output is equal to BG/P output*
This is one of the most important validations. The prerecorded data should be

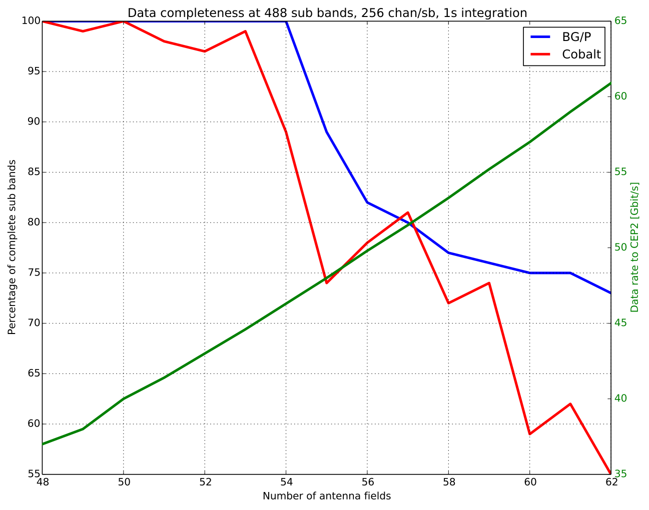

Figure 5.2: Fraction of sub bands without any data loss in the final Measurement Sets.

fed through both Cobalt *and* the BG/P. The output data sets are subsequently subtracted. Because the Fourier transforms in Cobalt may at some point all be normalized to ensure that the flux scale is independent of the number of channels, one may need to apply a constant multiplication factor between Cobalt and the BG/P before subtracting the data sets.

Any deviations must be minor and consistent with numerical noise, expected to be at a level of about 10^{-5} of the visibility amplitude. The difference must also be un-biased. If Cobalt passes this test, we have conclusively shown that Cobalt is no worse than the BG/P.

Because the BG/P apparently can not digest offline UDP packets anymore, we cannot do this experiment and will have to independently verify Cobalt's functionality and performance.

Exp. 16 [PASSED 2014-03-13]: Correlator capacity [real time data]
Aim: *Find the maximum correlator capacity*

In 8 bit mode with 488 sub bands, in HBA_DUAL configuration, with 256 channels per sub band and 1 second integration time, conduct a series of 1 minute observations in which the number of correlated fields is increased every time until the correlator can not keep up anymore.

Estimate, from the processing power / network capacity that is used as a function of the number of stations, at which point the correlator can not keep up anymore. Once all stations are being correlated, one can try to decrease the integration time in steps of a factor of two until data writing breaks down, to establish the maximum obtained throughput to CEP2.

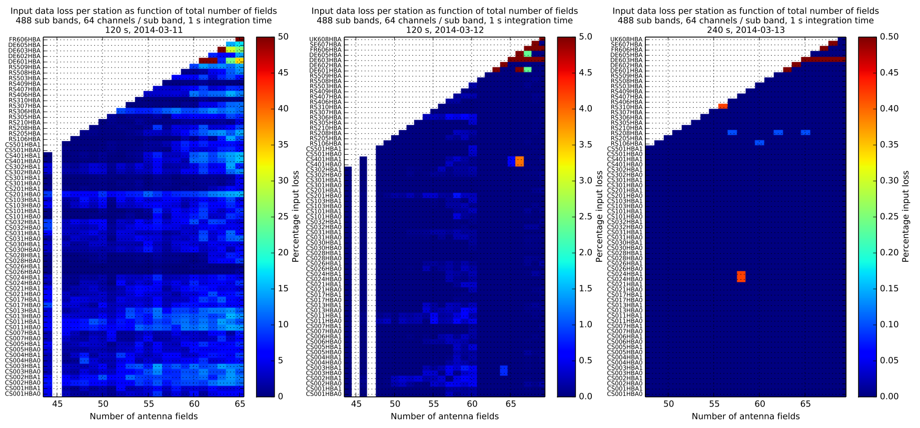

Figure 5.3: Data loss at the input of Cobalt per antenna field as a function of the total number of antenna fields in the observation. These are the statistics for three days in a period in which we were finalizing the input optimizations. Note the scale of the colorbars. It goes down by a factor 10 each day.

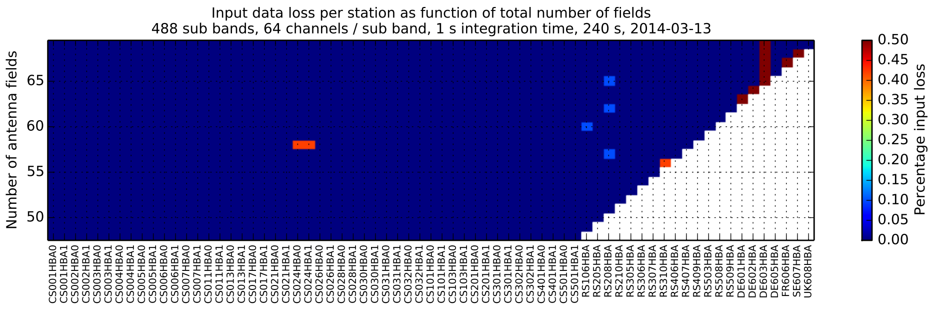

Figure 5.4: Data loss at the input of Cobalt per antenna field as a function of the total number of antenna fields in the observation. This is an enlargement of the third panel in Fig. 5.3

We tested 48 to 62 antenna fields (2014-01-29). SAS IDs 202756..202772. We find that Cobalt's percentage of complete data sets (Fig. 5.2) roughly follows the shape of the BG/P's curve, but looses a bit more, and behaves more erratically. At the data rates of typical observations (4 Gbit/s), however, we do not loose data. Although the erratic behaviour must be solved, the performance is adequate for regular operations.

We also tested the input loss before and after optimizing the input section of Cobalt, after pretty much having rewritten the entire initial data distribution before correlating. We observed in 8 bit mode, using 488 sub bands, 64 channels per sub band, and 1 s correlator integration time. The result is shown in Fig. 5.3. We have shown that we typically loose less than one percent of data per station on 120 s long observations. Most loss is incurred in the first 5–15 seconds. The international stations drop way more data, but those data simply do not arrive at Cobalt due to issues in the long haul network connections towards those stations. These connection problems have been resolved after this experiment was conducted. The left hand graph is before basic optimization, the right hand graph is the current situation. That panel is enlarged in Fig. 5.4.

Exp. 17 [PASSED 2014-03-14]: Long, deep integration [real data (12h integration on 3C 48 or 3C 147)]
Aim: *Establish long term stable operation of Cobalt*
Observe 12 h of 3C 48 HBA_DUAL_INNER data in 8-bit mode with all Dutch LOFAR stations. This is basically an EoR type experiment. Flag the data and calibrate with the best source model there is for this field, obtainable from V. Pandey. If we reach the thermal noise on the inner 8 h of data – expected to be around 100–150 μJy/beam. – and there are no artifacts visible in the residual map, the correlator pipeline is good enough.

We have taken a 5h EoR type observation of 3C 196, which showed correlator capacity problems that had appeared during the 2014-02-03 stop day. These data curently reside on the EoR cluster and still need to be imaged.

On 2014-03-13, we performed an 11h 8 bit observation of 3C 196 with all stations, except DE604, in HBA_DUAL_INNER mode. That is, we used 69 antenna fields. The observation ran fine and lost no data at the output. Input losses were benign, and most likely due to data not arriving at Cobalt in the first place. The only stations that lost 0.01 % or more of their input data were:

CS004HBA0	0.01%
CS004HBA1	0.01%
CS401HBA0	0.03%
CS401HBA1	0.03%
DE601HBA	2.65%
DE602HBA	0.06%
DE603HBA	32.57%
DE605HBA	2.63%

FR606HBA 2.64%

SE607HBA 38.98%

UK608HBA 2.52%.

The input losses were likely due to data that never even arrived at Cobalt.

The data are currently [2014-03-14] at the EoR cluster, and will be imaged by V. Pandey.

Chapter 6

Beamformed modes

Exp. 18 [IN PROGRESS]: Coherent stokes [recorded data: PSR B0329+54]
Aim: *Verify that coherent addition increases SNR linearly*
Take the PSR B0329+54 test data sets, and coherently add 1, 2, 4, 8, 16, 32, and 46 antenna fields. Do this in 8-bit and 16-bit mode. The SNR increase must be linear in the number of stations, and the flux scale the same in both bit modes.

The data must be processed using the pulsar pipeline and the PRESTO package. Repeat this in Stokes I, $IQUV$, and for complex voltages. Use 1, 16, and 64 channels.

We have shown that coherent and incoherent addition work in offline mode in the daily image of [2014-02-14] (see Fig 6.1). On-line addition works too, however, as Fig. 6.2 shows, its signal to noise ratio (SNR) falls well below the theoretically expected curve. The cause may be the station-delay calibration table at Cobalt, something at the stations, local interference, or a problem in Cobalt's addition kernels. These things are currently (2014-08-29) under investigation.

Exp. 19 [IN PROGRESS]: Incoherent stokes [recorded data: PSR B0329+54]
Aim: *Verify that incoherent addition increases the SNR* $\propto \sqrt{N}$
. . . where N is the number of stations. Take the PSR B0329+54 test data sets, and coherently add 1, 2, 4, 8, 16, 32, and 46 antenna fields. Do this in 8-bit and 16-bit mode. The SNR increase must be proportional to the square root of the number of stations, and the flux scale must be the same in both bit modes.

The data must be processed using the LOFAR pulsar pipeline. Repeat this in Stokes I, $IQUV$, and for complex voltages. Use 1, 16, and 64 channels.

We have shown that coherent and incoherent addition work in offline mode in the daily image of [2014-02-14] (see Fig 6.1). On-line addition works too, however, as Fig. 6.2 shows, its signal to noise ratio (SNR) falls well below the theoretically expected curve. The cause may be the station-delay calibration table at Cobalt, something at the stations, local interference, or a problem in Cobalt's addition kernels. These things are currently (2014-08-29) under investigation.

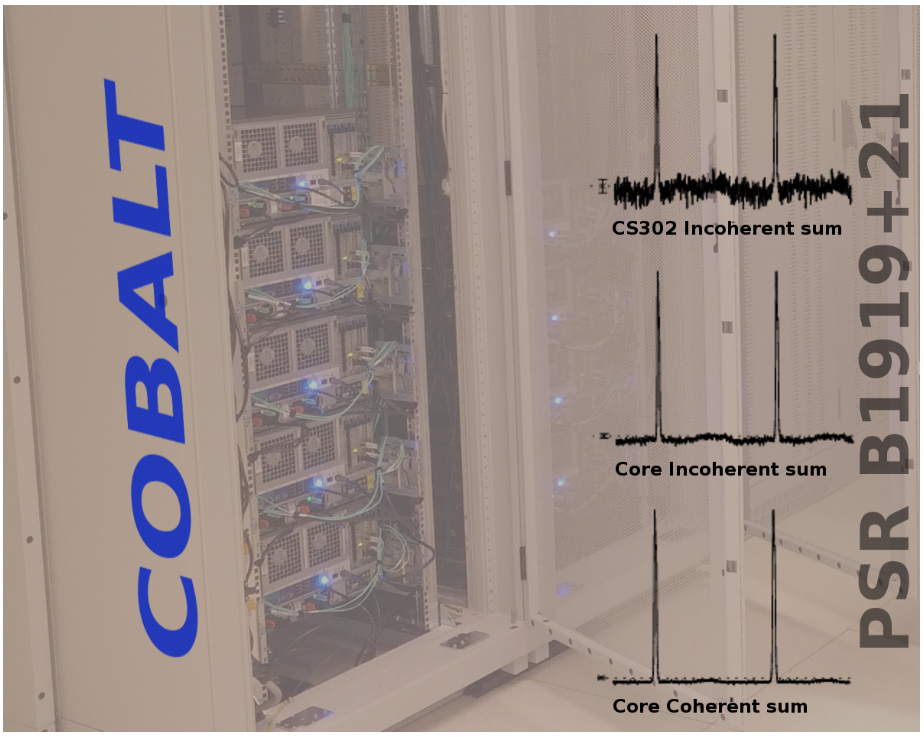

Figure 6.1: Detection of LGM-1 with incoherent and coherent beam forming by Cobalt in offline mode.

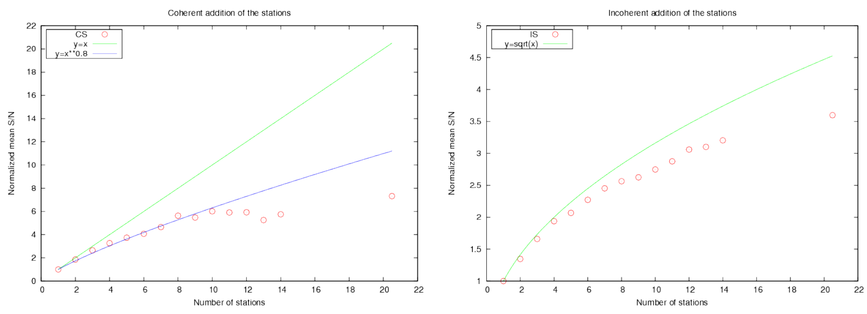

Figure 6.2: Coherent Stokes (left) and incoherent Stokes (right) signal-to noise ratio as a function of the number of stations in the (in)coherent sum. The SNR falls well below the theoretically optimal (green) curves. Cause is being investigated.

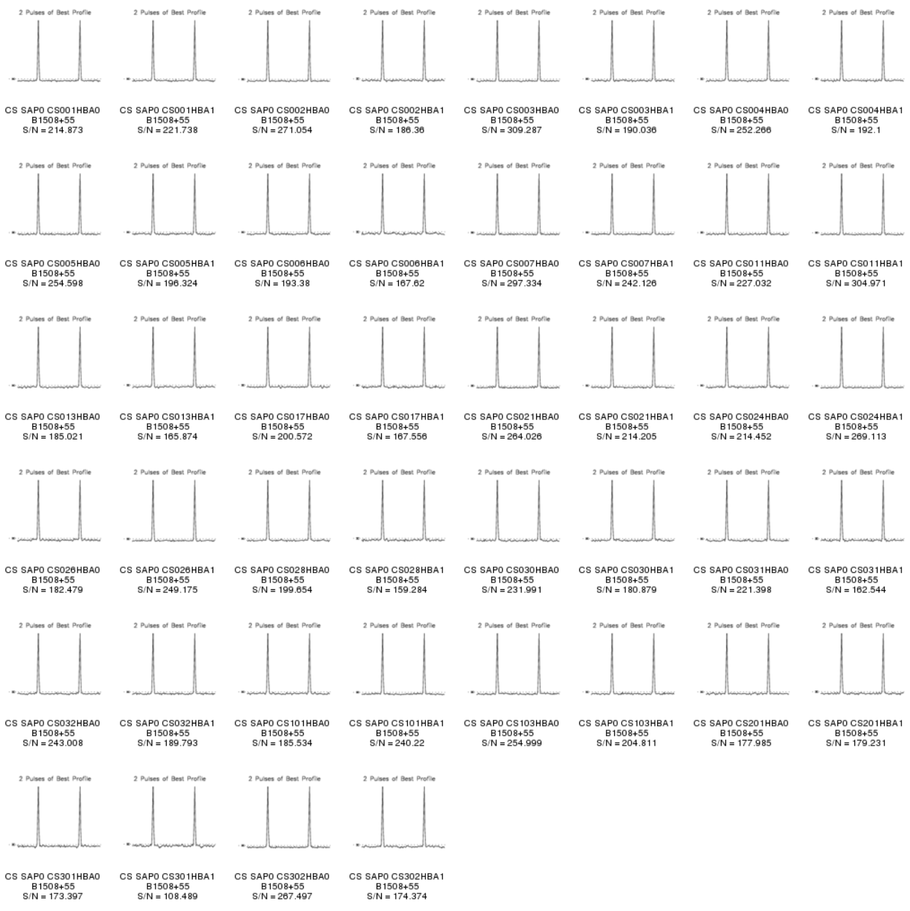

Figure 6.3: One of the first successful poor man's Fly's Eye observations with Cobalt.

Exp. 20 [PASSED 2014-04-24]: Fly's eye [recorded data: PSR B0329+54]
Aim: *Ensure we can separately record data from individual stations*
Set up parallel observations of 1, 2, 4, 8, 16, 32, and 46 antenna fields. process the resulting data and recover the pulsar SNR per station. Do this only in 16 bit mode for Stokes $IQUV$.

We have done several poor-man's fly's eye observations (all stations looking in the same direction), which is a useful system debugging tool. This seems to work d.d. [2014-03-13] for 69 antenna fields in 16-bit mode. Proper fly's eye observing (every station observing a different direction) is not yet possible [2014-03-14].

In fact, this can be done without resorting to parallel observations. Specifying "poor-man's Fly's Eye" observations (all stations pointing in the same direction) works the same way as with BG/P. See Fig. 6.3 for first results. A full fly's eye observation (all stations in independent directions) must (and can) be specified by multiple parallel observations.

Exp. 21 [CANCELLED]: Equivalence to BG/P in 16 bit mode [recorded data: PSR B0329+54]
Aim: *Assert that BG/P and Cobalt data are equivalent*
The prerecorded data should be fed through both Cobalt *and* the BG/P. The output data sets are subsequently subtracted. Because the Fourier transforms in Cobalt may at some point all be normalized to ensure that the flux scale is independent of the number of channels, one may need to apply a constant multiplication factor between Cobalt and the BG/P before subtracting the data sets.

Any deviations must be minor and consistent with numerical noise, expected to be at a level of about 10^{-5} of the timeseries amplitude. The difference must also be un-biased. If Cobalt passes this test, we have conclusively shown that Cobalt is no worse than the BG/P.

Exp. 22 [TODO]: Coherent dedispersion [synthetic and recorded data]
Aim: *Verify that coherent dedispersion works in HBA and LBA.*
Using synthetic data of a highly dispersed pulse, verify that correcting for the exact dispersion measure yields the shape and amplitude of the pulse that went in.

Using prerecorded station data of a highly dispersed pulsar, perform coherent stokes beam forming, applying coherent dedispersion, and verify that the signal-to-noise ratio and pulse profile in the output correspond with what the pulsar group expects.

This functionality will be implemented during regular operations.

Exp. 23 [PASSED]: Coherent beam forming capacity [real time data]

Aim: *Find the maximum number of coherent beams*
In 8 bit mode with 488 subbands, in HBA_DUAL configuration, with 16 channels per sub band and no time integration, conduct a series of 5 minute coherent stokes observations in which the number of tied array beams is increased every time until the correlator can not keep up anymore. Beam form the entire core in HBA_DUAL mode with the following tied array beam numbers: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 and 17, 31, 67, 127, 257, 509.

Do not use coherent dedispersion.

We have been probing the maximum capacity using tied array beam rings. We can do 5 tied array rings with 488 sub bands and all superterp HBA fields, loosing about 2% of data in that case. 4 tied array rings plus 12 separate tied array beams can be done with 37 Gbit/s data output rate, dropping only 0.8% of the data enroute to CEP2.

For the full core, we can do 6 tied array beams with 162 sub bands each. Beamformer performance is a function of approcimately 7 variables, making this a very hard parameter space to fully explore. We can computationally support al types of pulsar observations that were requested for LOFAR Cycle 2.

Exp. 24 [TODO]: Coherent dedispersion capacity [real time data]
Aim: *Find the maximum dispersion that can be handled in a single tied array beam*
For both LBA and HBA_DUAL, in 8-bit mode with 488 sub bands of 16 channels each, and one tied array beam, do coherent dedispersion, increasing the dispersion measure in factors of 2, beginning at 1, until Cobalt runs out of memory.

Coherent dedispersion will be implemented later.

Exp. 25 [IN PROGRESS]: Long, deep integration [real time data]
Aim: *Establish long term stability*
Observe a weak pulsar and its environment using the maximum number of tied array beams in 8 bit mode with 46 antenna fields for several hours. The system may not crash, and must yield thermal noise limited data.

The beamformer is stable over several hours of observing. However, the coherent stokes sensitivity with the core is currently (2014-08-29) subpar. Cause is yet unknown. It may be the delay calibration table, something in the coherent stokes kernel, or something with the stations themselves. This is actively being investigated.

Exp. 26 [PASSED 2014-05-28]: Millisecond pulsars [real time data]
Aim: *Establish absolute time*
Observe a group of known millisecond pulsars to find out if the time scale in the output files is reproducably close enough to UTC.

Timing residuals on J0034-0534 are approximately 24 μs RMS and comparable to BG/P era. See Fig. 6.4.

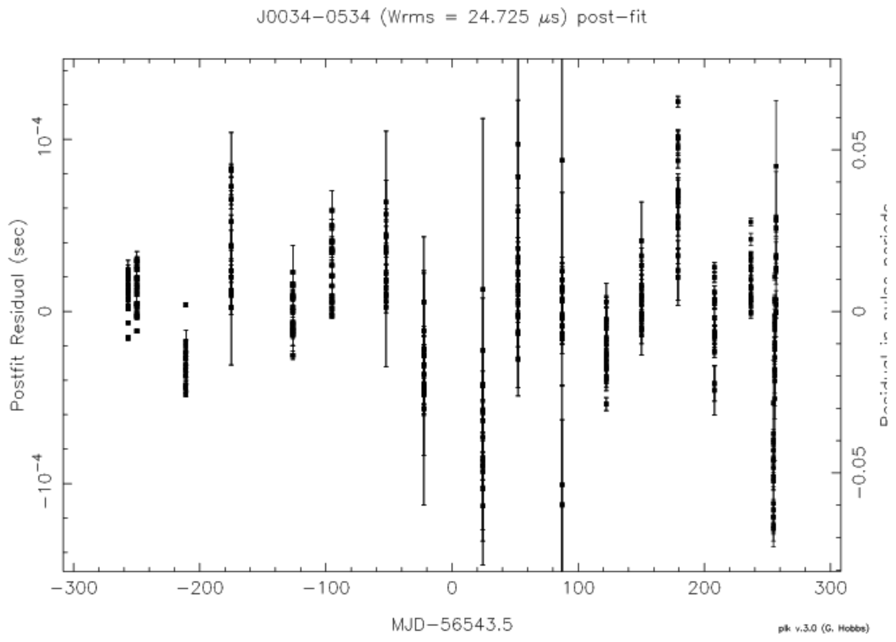

Figure 6.4: Timing residuals on J0034-0534 after jointly fitting BG/P and Cobalt data.

Chapter 7

Parallel observations

Exp. 27 [PASSED 2014-06-17]: Multiple simultaneous identical pipelines [real time data]
Aim: *Ensure we can run multiple simultaneous pipelines at all*
Cobalt's fly's eye mode will be implemented with multiple parallel observations where each antenna field take part in at most one observation. This is obviously the simplest case.

In this experiment, run multiple simultaneous coherent stokes observations of the same pulsar in 8-bit HBA_DUAL mode. Every observation uses one antenna field. Use progressively more antenna fields in approximate powers of two:

- superterp stations (6 stations, 12 fields)
- half of the core stations (12 stations, 24 fields)
- all core stations (24 stations, 48 fields)
- all Dutch stations (38 stations, 62 fields)
- all stations (46 stations, 70 fields)

Repeat this experiment for the correlator pipeline by executing parallel observations with the core only, remote stations only, and international stations only.

We have, technically, not strictly tried this experiment, but discovered on June 17, that we can schedule and run arbitrary observations simultaneously, using different start and stop times, provided that the station sets do not overlap.

Exp. 28 [IN PROGRESS]: Multiple different, simultaneous pipelines, overlapping data [real time data]
Aim: *Run interferometric and beam formed observations at the same time*
Produce correlated data and beam formed data from the *same* set of stations.

Code is ready for commissioning as of 2014-08-29.

Exp. 29 [PASSED 2014-06-17]: Truly independent scheduling [real time data]
Aim: *Independently schedule and run different observations*
Run a set of independent observations in different beam forming modes and interferometric modes, that may not start and end at the same time.

This is mostly a long term wish, testing the observatory software more that Cobalt itself.

It works! On June 17, we scheduled and ran arbitrary observations simultaneously, using different start and stop times, provided that the station sets do not overlap. This possibility has since been used regularly for solar observations and interplanetary scintillation experiments.

Chapter 8

Observatory software integration

Exp. 30 [PASSED 2013-11-01]: Switch data streams to Cobalt or BG/P [real data]
Aim: *Ensure easy switching between the two correlators*
The switching is done by regenerating the configuration files for the stations. This is done by the createFiles script. This works beautifully for the Dutch stations. Switching between BG/P and Cobalt operations takes about 2 minutes now.

International stations still have routing issues, but I consider that separate from this experiment. In fact, most of those issues have been solved on 2013-11-26/27, and the final remaining connection issues to the international stations were resolved at the end of spring 2014.

Exp. 31 [PASSED 2014-03-14]: System validation observations [real data]
Aim: *Verify seamless integration into observatory software*
Run the standard suite of system validation observations from MoM through the Scheduler until all validation plots have appeared. At no point should it be necessary to specify anything Cobalt specific. This assumes the switch to Cobalt has already been made by the script used in the first experiment.

Finally, export the raw data of this validation run to the LTA.

On 2014-03-14 we have performed a successful system validation run. We have also ingested raw data from a previous, incomplete run into the LTA. The data and metadata of those observations appeared fine in the LTA.

Exp. 32 [PASSED 2014-04-24]: CEP-2 pipeline processing [real data]
Aim: *Verify that current processing pipelines are not broken*
Run observations with attached pipelines. Observations need not be longer than

10–20 minutes. Use the following combinations:

- LBA_INNER, flagging and demixing [PASSED 2014-01-27]
- HBA_DUAL_INNER, flagging and averaging [PASSED 2014-01-27]
- HBA_DUAL, target/calibrator pipelines
- HBA_DUAL_INNER, MSSS observation all the way to automatic imaging

Finally, export the processed data to the LTA.

Raw data as well as processed data has been exported to the LTA successfully. We run all pipelines that ran for BG/P data successfully in production as of 2014-04-24.

Chapter 9

Conclusions

Cobalt is sufficently well tested to use in regular observations, although some issues remain, primarily in the beam former pipeline. The highest priority issues to solve in the near future are

- poor scaling of coherent- and incoherent Stokes sensitivity as a function of number of input stations;
- residual station band pass in beam formed modes;
- delay model errors, particularly affecting long baselines.

Appendix A

verify-ms-format.py

```
#!/usr/bin/env python2

from pyrap.tables   import table
from pyrap.measures import measures
from pyrap.quanta   import quantity
from numpy import array, pi, arccos, inner, concatenate, unique
from numpy.linalg import norm


import sys


def test_ant_columns(msname):
    r'''
␣␣␣␣Test␣if␣ANTENNA1␣<=␣ANTENNA2.
␣␣␣␣'''
    tab  = table(msname)
    ant1 = tab.getcol('ANTENNA1')
    ant2 = tab.getcol('ANTENNA2')

    ok = ant1 <= ant2
    if not all(ok):
        raise ValueError('ANTENNA1␣is␣not␣always␣␣<=␣ANTENNA2')
    return ok




def test_uvw_casacore(msname):
    r'''
␣␣␣␣Test␣if␣UVW␣=␣UVW_FROM_ITRF(XYZ_ANTENNA2␣-␣XYZ_ANTENNA1)
␣␣␣␣'''

    dm = measures()

    tab = table(msname)
    ant1 = tab.getcol('ANTENNA1')
    ant2 = tab.getcol('ANTENNA2')
    uvw  = tab.getcol('UVW')
    time = tab.getcol('TIME')
    time_epochs = [dm.epoch('UTC',quantity(mjds, 's'))
                   for mjds in concatenate([time[:3000], time[-3000:]], axis=0)]

    field_table          = table(tab.getkeyword('FIELD'))
    phase_dir            = field_table.getcol('PHASE_DIR')[0,0,:]
    phase_dir_meas_info  = field_table.getcolkeyword('PHASE_DIR', 'MEASINFO')
    phase_dir_units      = field_table.getcolkeyword('PHASE_DIR', 'QuantumUnits')
    phase_dir_quantities = ['%r%s' % (angle, unit)
                            for angle, unit in zip(phase_dir, phase_dir_units)]
    lofar = dm.position('ITRF', '3826577.462m', '461022.624m', '5064892.526m')
    dm.do_frame(lofar)
    dm.do_frame(dm.direction(phase_dir_meas_info['Ref'],
                             phase_dir_quantities[0], phase_dir_quantities[1]))

    ant_table = table(tab.getkeyword('ANTENNA'))
    xyz       = ant_table.getcol('POSITION')
    xyz_1 = array([xyz[ant,:] for ant in ant1])
    xyz_2 = array([xyz[ant,:] for ant in ant2])
    delta_xyz = (xyz_2 - xyz_1)

    uvw_computed = []
    for dxyz, epoch  in zip(delta_xyz[:2000], time_epochs):
        dm.do_frame(epoch)
        bl_quant = [quantity(value, 'm') for value in dxyz]
        #print bl_quant
        bl = dm.baseline('ITRF', bl_quant[0], bl_quant[1], bl_quant[2])
        #print bl
        uvw_computed.append(dm.to_uvw(bl))

    for a1, a2, uvw_ms, uvw_comp in zip(ant1, ant2, uvw, uvw_computed):
        uvw_comp = array(uvw_comp['xyz'].get_value('m'))
        if norm(uvw_ms) > 0.0 :
            arg = inner(uvw_ms, uvw_comp)/(norm(uvw_ms)**2)
            if arg > 1.0:
                arg = 1.0
            if arg < -1.0:
                arg = -1.0
            angle_mas = arccos(arg)*180*3600*1000./pi
            if angle_mas > 50.0:
                fmt = 'Angle␣UVW␣computed␣/␣MS␣%03d--%03d␣=␣%.3f␣mas␣(%.3f␣deg)'
                raise ValueError(fmt %
                                 (a1, a2, angle_mas, angle_mas/1000./3600.0))
        diff = uvw_ms - uvw_comp
        if norm(diff) > 1e-3:
            raise ValueError('UVW␣computed␣-␣UVW␣MS␣%03d--%03d␣=␣%.3f␣mm' %
                             (a1, a2, norm(diff)*1e3))

    return True



def test_uvw_new_casacore(msname):
    r'''
␣␣␣␣Test␣if␣UVW␣=␣UVW_FROM_ITRF(XYZ_ANTENNA2␣-␣XYZ_ANTENNA1)
␣␣␣␣'''

    dm = measures()

    tab = table(msname)
    ant1 = tab.getcol('ANTENNA1')
    ant2 = tab.getcol('ANTENNA2')
    uvw  = tab.getcol('UVW')
    time = tab.getcol('TIME')
    time_epochs = [dm.epoch('UTC',quantity(mjds, 's'))
                   for mjds in concatenate([time[:3000], time[-3000:]], axis=0)]

    field_table          = table(tab.getkeyword('FIELD'))
    phase_dir            = field_table.getcol('PHASE_DIR')[0,0,:]
    phase_dir_meas_info  = field_table.getcolkeyword('PHASE_DIR', 'MEASINFO')
    phase_dir_units      = field_table.getcolkeyword('PHASE_DIR', 'QuantumUnits')
    phase_dir_quantities = ['%r%s' % (angle, unit)
                            for angle, unit in zip(phase_dir, phase_dir_units)]


    lofar = dm.position('ITRF', '3826577.462m', '461022.624m', '5064892.526m')
    dm.do_frame(lofar)
    dm.do_frame(dm.direction(phase_dir_meas_info['Ref'],
                             phase_dir_quantities[0], phase_dir_quantities[1]))

    ant_table = table(tab.getkeyword('ANTENNA'))
    xyz       = ant_table.getcol('POSITION')
    xyz_1 = array([xyz[ant,:] for ant in ant1])
    xyz_2 = array([xyz[ant,:] for ant in ant2])
    delta_xyz = (xyz_2 - xyz_1)

    uvw_computed = []
    for ant1_xyz, ant2_xyz, epoch  in zip(xyz_1[:2000], xyz_2[:2000], time_epochs):
        dm.do_frame(epoch)

        ant_1_quant = [quantity(p, 'm') for p in ant1_xyz]
        ant_2_quant = [quantity(p, 'm') for p in ant2_xyz]
        ant_pos = dm.position('ITRF', map(list, zip(ant_1_quant, ant_2_quant)))

        bl_quant = [quantity(value, 'm') for value in dxyz]
        #print bl_quant
        bl = dm.baseline('ITRF', bl_quant[0], bl_quant[1], bl_quant[2])
        #print bl
        uvw_computed.append(dm.to_uvw(bl))

    for a1, a2, uvw_ms, uvw_comp in zip(ant1, ant2, uvw, uvw_computed):
        uvw_comp = array(uvw_comp['xyz'].get_value('m'))
        if norm(uvw_ms) > 0.0 :
            arg = inner(uvw_ms, uvw_comp)/(norm(uvw_ms)**2)
            if arg > 1.0:
                arg = 1.0
            if arg < -1.0:
                arg = -1.0
            angle_mas = arccos(arg)*180*3600*1000./pi
            if angle_mas > 50.0:
                fmt = 'Angle UVW computed / MS %03d--%03d = %.3f mas (%.3f deg)'
                raise ValueError(fmt %
                                 (a1, a2, angle_mas, angle_mas/1000./3600.0))
        diff = uvw_ms - uvw_comp
        if norm(diff) > 1e-3:
            raise ValueError('UVW computed - UVW MS %03d--%03d = %.3f mm' %
                             (a1, a2, norm(diff)*1e3))

    return True



def test_frequency_labels(msname):
    tab = table(msname)
    spw_tab = table(tab.getkeyword('SPECTRAL_WINDOW'))
    chan_freq = spw_tab.getcol('CHAN_FREQ')
    ref_freq  = spw_tab.getcol('REF_FREQUENCY')
    num_chan  = spw_tab.getcol('NUM_CHAN')
    sb_bandwidth = spw_tab.getcol('TOTAL_BANDWIDTH')

    clock_mhz = None
    if sb_bandwidth[0] == 100e6/512.:
        clock_mhz = 200
    if sb_bandwidth[0] == 80e6/512.:
        clock_mhz = 160
    if not clock_mhz in [160, 200]:
        raise ValueError('Strange sub band width(s) %r' % sb_bandwidth)
    if len(unique(sb_bandwidth)) != 1:
        raise ValueError('Multiple different sub band widths: %r' % sb_bandwidth)

    for i, (freqs, ref) in enumerate(zip(chan_freq, ref_freq)):
        if len(freqs) != num_chan[i]:
            raise ValueError('SB %d: # channel frequencies (%r) != NUM_CHAN (%r)' %
                             (i, len(freqs), num_chan[i]))
        if freqs[num_chan[i]/2] != ref:
            raise ValueError('SB %d: chan_freq[%d](%r) != ref(%r)' %
                             (i, num_chan[i]/2, freqs[num_chan[i]/2], ref))
        chan_width = unique(freqs[1:]-freqs[:-1])
        if len(chan_width) != 1:
            raise ValueError('SB %d: Channel frequencies are not equidistant' % i)
        if chan_width <= 0.0:
            raise ValueError('SB %d: Channel frequencies are in reversed order' % i)
        if abs(chan_width[0] - sb_bandwidth[i]/num_chan[i]) > 1e-3:
            fmt = 'SB %d: Channel frequency spacing != sub band width / num_chan'
            raise ValueError(fmt % i)
    obs = table(tab.getkeyword('OBSERVATION'))
    if clock_mhz != int(obs[0]['LOFAR_CLOCK_FREQUENCY']):
        raise ValueError('clock_mhz(%r)␣!=␣OBSERVATION.LOFAR_CLOCK_FREQUENCY(%r)' %
                         (clock_mhz, int(obs[0]['LOFAR_CLOCK_FREQUENCY'])))
    filter_name = obs[0]['LOFAR_FILTER_SELECTION']
    if filter_name in 'LBA_10_90␣LBA_30_90':
        if ref_freq.max() >= 100e6:
            raise ValueError('%r␣MHz␣not␣in␣%s' % (ref_freq.max()/1e6, filter_name))
    elif filter_name == 'HBA_110_190':
        if ref_freq.min() < 100e6 or ref_freq.max() > 200e6:
            raise ValueError('%r␣or␣%r␣MHz␣not␣in␣%s' %
                             (ref_freq.moin()/1e6, ref_freq.max()/1e6, filter_name))
    elif filter_name == 'HBA_210_250':
        if ref_freq.min() < 200e6 or ref_freq.max() > 300e6:
            raise ValueError('%r␣or␣%r␣MHz␣not␣in␣%s' %
                             (ref_freq.moin()/1e6, ref_freq.max()/1e6, filter_name))
    else:
        raise ValueError('I␣don\'t␣know␣what␣to␣do␣with␣filter␣%s' %
                         filter_name)
    return True




def test_uvw_calc(msname):
    raise NotImplementedError()


def test_autocorr_real(msname):
    tab = table(msname)
    antenna_table = table(tab.getkeyword('ANTENNA'))
    field_names = antenna_table.getcol('NAME')
    num_ant = len(field_names)
    fractional_eps = 1e-9
    ants_with_nonzero_imags = []
    for ant in range(num_ant):
        selection = tab.query('ANTENNA1␣==␣%d␣&&␣ANTENNA2␣==␣%d' % (ant, ant))
        data = selection.getcol('DATA')[:, 1:-1, 0::3]
        data = data.imag/data.real
        if abs(data).max() > fractional_eps:
            ants_with_nonzero_imags.append((field_names[ant], abs(data).max()))
    if len(ants_with_nonzero_imags) > 0:
        raise ValueError('Autocorrelations␣of␣%r␣has␣imaginary␣parts␣>␣%e*real!' %
                         (ants_with_nonzero_imags, fractional_eps))




def number_switched_off_rcus(field_name, rcu_flags):
    if 'HBA0' in field_name:
        return rcu_flags[:24,:].sum()
    if 'HBA1' in field_name:
        return rcu_flags[24:,:].sum()
    return rcu_flags.sum()


def test_antenna_field(msname):
    tab = table(msname)
    antenna_field_info = table(tab.getkeyword('LOFAR_ANTENNA_FIELD'))
    if antenna_field_info.nrows() == 0:
        raise ValueError('LOFAR_ELEMENT_FAILURE␣empty')
    element_flags = [row['ELEMENT_FLAG'] for row in antenna_field_info]
    antenna_table = table(tab.getkeyword('ANTENNA'))
    field_names = antenna_table.getcol('NAME')
    if len(element_flags) != antenna_table.nrows():
        raise ValueError('Length␣of␣LOFAR_ANTENNA_FIELD␣table␣not␣equal␣to␣ANTENNA')
    broken_rcus = array([number_switched_off_rcus(name, rcu_flags)
                         for name, rcu_flags in zip(field_names,
                                                    element_flags)])
    field_off_pairs = zip(field_names, broken_rcus)
    print('\n'.join(['%s:␣%d' % (name, off)
                     for name, off in field_off_pairs]))
    if any(broken_rcus % 2) == 1:
        raise ValueError('Only␣even␣numbers␣of␣broken␣RCUs␣expected')
    if broken_rcus.sum() == 0:
        raise ValueError('No broken RCUs detected?')
    no_broken = [off == 0 or ('RS' in field and 'HBA' in field and  off == 48)
                 for field, off in field_off_pairs]
    if all(no_broken):
        raise ValueError('Only 0 or 48 antenna\'s off? UNLIKELY!')
    return True



def run_tests(test_cases, msname):
    successful = []
    failed     = []
    for test in test_cases:
        try:
            print 'Running %s' % test.__name__
            result = test(msname)
            successful.append(test.__name__)
        except:
            print '--- FAILURE ---'
            message = ('%s: %s: %s' %
                       (test.__name__,
                        sys.exc_info()[0].__name__,
                        sys.exc_info()[1]))
            print message+'\n'
            failed.append(message)
    print '\nSuccessful\n----------\n  %s' % '\n  '.join(successful)
    if len(failed) > 0:
        print '\n'
        print 'FAILED\n======\n\n  %s' % '\n  '.join(failed)
    else:
        print 'OK'
    return len(failed) == 0



if __name__ == '__main__':
    msname = sys.argv[1]
    print 'Verifying %s' % msname
    sys.exit(run_tests([eval(name) for name in dir() if name[0:5] == 'test_'],
                       msname))
```

Bibliography

Broekema, C. 2013, COBALT top level hardware design, Tech. rep., Astron

Broekema, C., Mol, J. D., & Nijboer, R. 2013, COBALT requirements and specifications, Tech. rep., Astron

Mol, J. D. 2013, COBALT top-level software design, Tech. rep., Astron