LOFAR calibration strategy

J.E.Noordam, ASTRON

14th November 2005

Abstract

This is the state of affairs at the LOFAR Critical Design Review, subsystem Calibration, 16/17 November 2005, Dwingeloo.

1 Introduction

LOFAR calibration is defined somewhat unconventionally as *the art of completely subtracting the many foreground sources from a LOFAR observation* (see fig 1). The assertion is that full removal is possible only if the utilised Measurent Equation (M.E.) is correct, and if the values of its parameters are known accurately. In other words, if the instrument is fully calibrated. Some of these sources will be (very) bright and complex, and many will be in the high sidelobes of the primary beam(s). The LOFAR calibration challenges can be summarized as follows:

- Pathological ionosphere (up to 1 rad change per 10 s, variable over FOV).
- Very crowded fields (causes confusion, expensive processing).
- Very bright sources (requires very high dynamic range).
- Variable (f,t) station beamshapes (expensive processing).
- Instrumental polarisation (different X/Y voltage beams)
- High station beam sidelobes (all-sky imaging)
- New types of instrumental effects (e.g. BSR)
- Variable PSF over the residual images (difficult to deconvolve).
- Huge data volume (allows minimal operations per data-point).

Calibrating LOFAR will not be easy, and will require a lot of processing power. Moreover, the existing calibration techniques will not be sufficient. For instance, traditional self-calibration only solves for *'uv-plane effects'*, with only one instrumental parameter per station, assumed to be valid for the entire field-of-view.

Figure 1: *Successful LOFAR calibration is defined somewhat unconventionally as the complete subtraction of the many foreground sources. It should be noted that this will probably be impossible for the most challengeing LOFAR project, the detection of the EoR signature. The best we can do in that case is to make sure that the signature of the source subtraction residuals is well understood, and sufficiently orthogonal to the expected EoR signal.*

By contrast, LOFAR will have substantial position-dependent *'image-plane effects'* caused by the ionophere and individually variable station beamshapes. This will make it necessary to solve for many more instrumental parameters per station, at considerable cost in processing[1]. Moreover, since the LOFAR point-spread-function (PSF) will vary significantly over the FOV, all known sources will have to be subtracted from the uv-data, while it will be difficult to deconvolve the rest from the residual images.

On the positive side, there is enough information available, in the form of bright calibrator sources, to solve the LOFAR calibration problem in principle[2], at least for a significant part of the time. See fig 2. Unfortunately, this will require significant processing per data-point, which will be a continuing challenge. *Therefore, it is imperative to set up the LOFAR calibration in such a way that the development of alternative approaches is actively encouraged, and can be implemented relatively easily.*

[1] This process has become known under the name of 'peeling'. The term was coined originally (cite ...) for subtracting (peeling) the contributions of bright sources from the uv-data one by one, in decreasing order of brightness. However, it is now used to indicate the solving for position-dependent instrumental errors in general.

[2] The situation is different in optical interferometry through the athmosphere, where there is usually not enough information (i.e. photon events) available per isoplanatic patch (2 as), per relaxation time (10 ms).

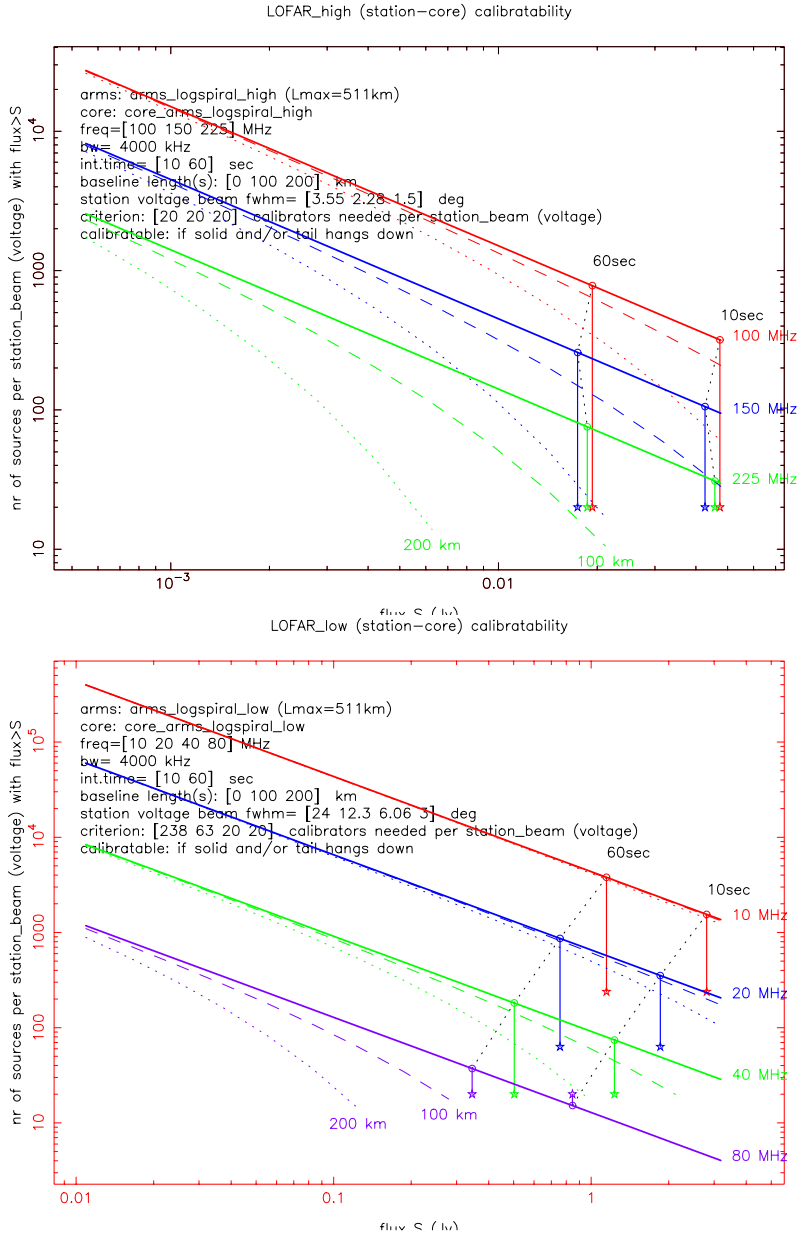


Figure 2: *The calibratability of LOFAR is determined by the availability of bright calibrator sources. The presence of enough information to solve for the M.E. parameters guarantees that LOFAR can be calibrated in principle, even if the precise calibration algorithm is not yet known.*
In these plots, which were made for the LOFAR PDR several years ago, the number of calibrators is sufficient if the vertical 'tail' hangs down. For the given instrumental parameters, this is always the case for point sources (solid lines), except for 10 s integration time at 80 MHz. For extended sources (dashed lines), the situations is less favorable, especially for longer baselines. This caused some concern at the time. But our thinking has evolved, and we now feel that the idea of solving for independent ionospheric phase screens per station, and our estimate of the necessary number of calibrators, was rather too severe. The Minimum Ionospheric Model (MIM, see section 11) requires very much fewer sources.

2 Development strategy

On the basis of experience with other large astronomical software projects elsewhere, the LOFAR Project has chosen for a calibration development strategy with two separate teams. This report is from the development team, which has the task to produce a *working prototype*. In a later stage, the implementation team will transfer the concepts to LOFAR hardware, using LOFAR software technology. The strategy for the development of the working prototype has the following elements:

1. To create the conditions for the (continuing) development of the LOFAR Measurement Equation (M.E.) by many people. Fortunately, the HBS formalism (cite ...) is essentially complete, i.e. it seems to cover all possible telescopes. However, its details need to be worked for the specific case of LOFAR, which will not be trivial.

2. To develop and test a first version of the LOFAR M.E. (MIM, station beamshapes).

3. To develop and prototype new processing concepts (e.g. peeling, LSM, uv-bricks)

4. To consider alternative methods where possible, and the trade-off function for cutting corners (e.g. source subtraction by filtering).

The vehicle chosen to realise these ambitious goals is the MeqTree module, which allows the rapid implementation of an arbitrary M.E., and is able to solve for (arbitrary subsets of) its many parameters.

For the immediate future, a sequence of WSRT observations has been selected in consultation with active astronomers, to exercise the various aspects of the working prototype:

- WSRT central point source (cps) calibration, including bandpass and polarisation. This exercises the handling (organisation, book-keeping, tools) of MeqParm tables. Ideally, this should be the start of the use of MeqTrees as part of WSRT operations.

- 3C343/3C343.1: This field has two dominant point sources, one of which is at the half-power point of the WSRT primary beam. This exercizes various 'peeling' issues, and dealing with a small number (20) of fainter sources. It also allows comparison with other packages, since this field has been reduced with MIRIAD, NEWSTAR, AIPS and AIPS++. In these packages, peeling is possible with considerable effort by experienced astronomers. Meqtrees make peeling much less laborious.

- Abell 965: Exercises the use of LSM patches and uv-bricks to predict an extended source and a largish group of fainter sources.

- Reduction of WSRT observations that include the experimental LOFAR/WHAT station. This exercises the modelling of LOFAR HBA voltage beams, and the combination of (highly) dissimilar stations. This is the first example of a project that cannot be handled by any of the existing packages.
- 3c147 @ 117 MHz (WSRT LFFE) exercises the Minimum Ionospheric Model (MIN) concept, and other low-frequency issues that are highly relevant for LOFAR. Optionally another LFFE field without a bright source could be attempted, to exercise solving for phase gradients only (after cross-calibration with a calibrator source). It could also be used to exercise redundant spacing calibration.
- 3c...: This field contains a substantial number of bright point sources in the field. It exrecises the solving for individual station beamshapes (which includes pointing errors).

See also the appendix.

3 Calibration = source subtraction

As stated before, calibration can be equated with the subtraction of foreground sources. Because of image-plane effects, they must be subtracted from the uv-data as much as possible. Unfortunately, in order to preserve the field size, subtraction must be done at full (f,t) resolution, which is expensive. We distinguish three categories of sources:

1. **Cat I sources** are the 20-50 sources per FOV that are so bright that they have to be subtracted individually. They are used for self-calibration, to estimate instrumental errors *in their specific direction*. These are used first of all to predict and subtract these bright sources with the greatest possible accuracy. But they are also used to estimate the parameters of the ionospheric phase function, and the station beamshapes, which are needed to subtract Cat II sources (see below). Cat I sources can be both in the main lobe (FOV) and the side-lobes of the station primary beams (see also section 12).
2. **Cat II sources** are the rest of the sources in the Local Sky Model (LSM). They are predicted and subtracted in *patches*, which are smallish regions in the sky. Cat II sources are NOT used for selfcal. They are identified and updated in residual images (see sections 7 and 10).
3. **Cat III sources** are too faint to be in the LSM, and therefore cannot be subtracted from the uv-data. They have to be dealt with in the image-plane somehow. For this reason, residual images must be made in the form of multiple smallish facets (see sections **??** and 10).

The dividing lines between the various source categories are fluid. If a Cat II source cannot be subtracted well, it is promoted to Cat I for individual treatment[3]. And as soon as a Cat III source has been identified and included into the LSM, it has become a Cat II source.

Bright (Cat I) point sources are predicted by DFT. Extended sources and Cat II sources are predicted by FFT of a gridded patch. The resulting cube of gridded uv-data is called a uv-brick (see section 8). Predicted values for individual ifrs are obtained by interpolation. *Very importantly, it turns out to be possible (and relatively cheap) to apply image-plane effects as part of this interpolation function.*

Obviously, sources can only be subtracted properly if the source parameters are known accurately. The parameters of Cat I sources may be estimated as part of the selfcal process, even if the sources are extended. The parameters of Cat II sources may be estimated by checking residual images for any residuals at the position where LSM sources have been subtracted.

4 Overview of the calibration system

The various blobs in the *Blob Diagram* of fig 3 represent the major elements of the proposed LOFAR calibration system. They are described in the context of the MeqTree system used for the working prototype, but the final LOFAR implementation will have the same components:

- **Global Sky Model (GSM):** Will contain *intrinsic* representations of all the sources that have been identified and updated over the lifetime of LOFAR. Mostly this will be in the form of parametrised source components (PSC), but very extended sources may be more conveniently represented by *GSM image cubes*, e.g. in the form of CLEAN components. Since LOFAR sources are too diverse to be constrained to a single parametrisation[4], we propose to include MeqTree subtrees into the GSM, as a convenient way to describe the relationship between the parameters of a source, and its manifestations (I,Q,U,V) in the two domains. In the same spirit, we propose to include funklets into the GSM, to represent parameter variability in frequency and time, and in other dimensions[5].

- **Local Sky Model (LSM):** Contains a subset of the GSM that is needed for a particular observation. It is used for various things like source prediction, source parameter update, GUI (!), process monitoring, etc. After

[3] We cling to the (as yet) unproved theorem that any source that is bright enough to cause trouble, is bright enough to be tackled somehow.

[4] Astronomical sources may be parametrised in many different ways. The minimum is three for the flux and position of an unpolarised point source, which does not vary in time or frequency. Extended sources may be described in terms of elliptic gaussians, but also as shapelets or pixons. The use of MeqTrees and funklets in the GSM allows them all.

[5] The inclusion of MeqTrees and funklets in an astronomical database represent a bit of a revolution. It might complicate the planned compatibility of the GSM with the Virtual Observatory. However, there is no alternative.

use, the updated LSM may be used to update the GSM. See section 7 for more details.

- **MeqTree kernel:** Used to implement an arbitrary Measurement Equation (in the form of MeqTrees), and to solve for its parameters. The latter is done by comparing measured uv-data with predicted values. It interfaces with an MS for measured uv-data (and flas and weights, and telescope info), and an LSM and instrumental MeqParm tables for calculating predicted values. The result is improved MeqParm values, and residual uv-data from which all LSM sources have been subtracted. The latter are written back to the MS. The MeqTree kernel has many powerful features for monitoring and controlling its operation, and for inspection of the results. These are important, but will not discussed here.

- **MeqParm tables:** Contain the values of M.E. parameters, e.g. station phase. The MeqParm tables for source parameters are part of the LSM. Each MeqParm has zero or more entries (rows) in a table, each of which contains a funklet and its validity domain (usually a rectangle in freq-time space). A funklet is an N-dimensional array of function coefficients, e.g. polynomials in freq and time. These coefficients are the real M.E. parameters, because they are the ones that are being solved for. There will be a range of MeqParm table tools to manipulate and visualise the available funklets per (group of) parameters.

- **Measurement Sets** (MS) are the AIPS++ uv-data holders. With a MeqTree interface layer they are sufficient for the purposes of the working prototype.

- **Imaging module:** AIPS++ contains a superset of the imaging modes in the existing packages, and this part is likely to be further developed in the future. It should be noted that we only re-use the imaging module in the narrow sense of converting (residual) uv-data into images. We do not subscribe to the wider sense of including selfcal, which is done by the MeqTree module.

- **Residual images:** The AIPS++ image class and related tools are sufficient for the purpose of the working prototype.

The above is just a reminder. The reader is assumed to be familiar with the overall structure of the selfcal process, and the MeqTree fundamentals (see also section 10). LOFAR deliverables will be the GSM, residual images, MeqParm tables and 'meta-data'.

Note the heavy re-use of AIPS++ modules. This has very considerable advantages, especially since AIPS++ has become much more stable in view of its central role in the ALMA project.

In the following, specific aspects of the calibration system will be treated in more detail.

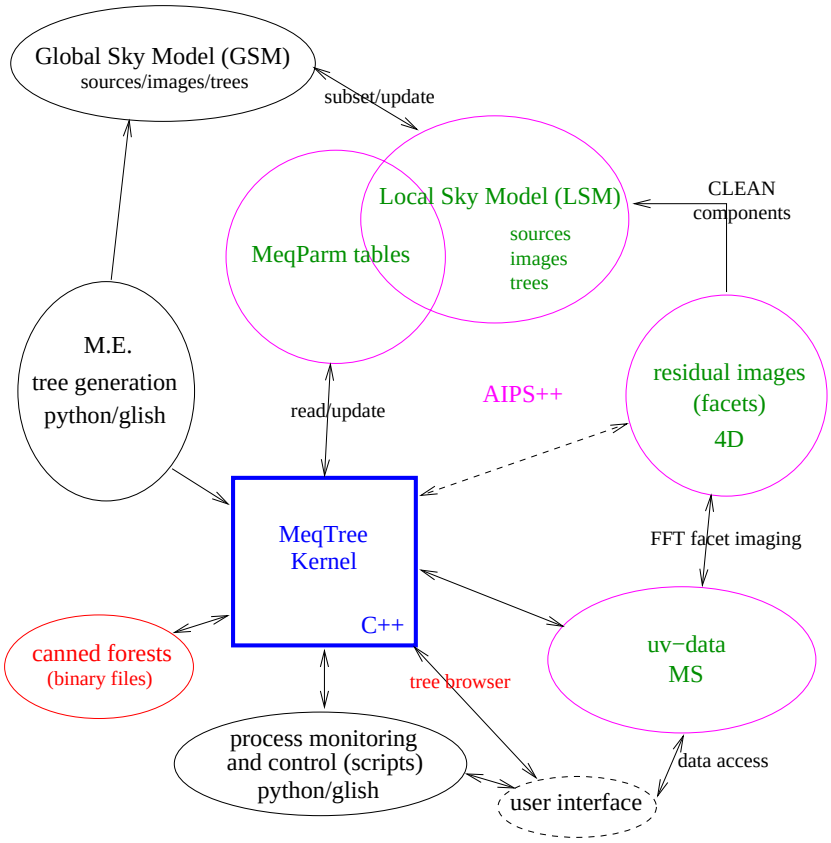


Figure 3: *The blob diagram shows the major components of the working prototype. Note the reuse of AIPS++ modules (MS, tables, images, FFT, fitting, functionals, measures, etc). Roughly speaking, the MeqTree kernel replaces the AIPS++ Calibrater module, and part of its Imager module. The members of the development team each work in their own area, so that they can be self-propelled without getting in each others way.*

5 Solving for M.E. parameters (MeqParms)

Arguably the most important function of the calibration system is solving for M.E. parameters. These are implemented as MeqParm nodes, which have no children, but have access to information that allows them to return values for any requested domain.

The values for an M.E. parameter (MeqParm) are stored in a MeqParm table, in the form of zero or more *funklets*, each of which has its own validity domain. In addition, a MeqParm is usually created with a default value. Thus, a MeqParm will *always* return a result vells, whatever the requested domain. When a MeqParm has been set *solvable* (by a MeqSolver, via its request), it also returns information that allows the solver to calcutate derivatives w.r.t. MeqParm funklet coefficients. In this stage, these are numerical derivatives[6]: for each coeff, an extra vellset is calculated with a small perturbation of the coeff value. Thus, the nodes downstream towards the MeqSolver have to perform their function on (much) more than one vellset, which is a potential bottleneck. Fortunately, some corners can be cut in this area (e.g. by using larger cells, see below), but we should also look out for alternative solver mechanisms.

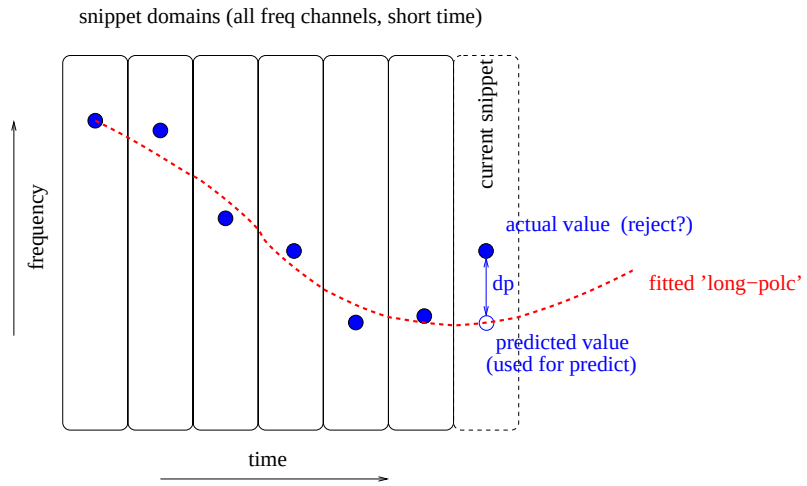
Usually, uv-data are processed in *snippets*, i.e. relatively small domains that cover an entire frequency band (up to several MHz), but only about a minute in time. This is done because many M.E. parameters (e.g. atmospheric phase) vary more smoothly in freq than in time. Since processing and (node cache) memory use are proportional to the number of domain cells, the cell size in the frequency direction should be maximised by combining groups of channels. After all, the minimum number of cells in a particular dimension is determined by the degree of the polynomial that is fitted in that dimension[7].

However, since there is some unavoidable overhead in node function calls, it is more efficient to go through the observation in a smaller number of domains that each cover more time. The optimum lies in *tiled solutions*, where a domain has a relatively large number of cells, but separate solutions are made for different tiles (sub-domains). This can be generalised in the sense that different parameters are solved for different tiles, using the same domain. However, if the tiles are one cell wide, the solution matrix is block-diagonal, which allows more efficient inversion.

Even if separate snippet solutions are made for a particular MeqParm, it is often desirable to build in some inertia, i.e. to make maximum use of its assumed continuity in time. This can be done in several ways. First of all, the solution funklets in the MeqParm table may be smoothed, i.e. multiple funklets

[6] Numerical derivatives have the advantage that they simple, and can be calculated through complex nodes like uv-bricks etc. If two perturbed vellsets are used per coeff, the resulting *double derivatives* may cause more rapid convergence, but the extra cost in processing and memory use (caches) may be prohibitive. In the future, we will consider the relative merits of analytical derivatives.

[7] Note that maximising the cell size is allowed in the case where it is most needed, i.e. in selfcal where lots of perturbed vellsets have to be calculated for derivatives. By contrast, source subtraction requires predicted values at full (t,f) resolution, but derivatives are not needed.



An elegant scheme for MeqParm solution behaviour:

- The blue dots (●) are the results of snippet solutions (c0f only)
- Each provides an equation that is added to the 'long−polc' solution matrix
- The latter is inverted each time, to give better long−polc coefficients
- The validity domain of the current long−polc is increased with each snippet
- The updated long−polc is used to predict the value for the current (next) snippet
- The differences (dp) between predicted and actual snippet values are kept
- These can be used to accept/reject a snippet solution in a policy−free manner
- They can also be used to decide to start a new long−polc

Question: Is this scheme equivalent to a single long−snippet solution for a long−polc?

Figure 4: *Schematic overview of (some aspects of) solving for Meqparm values (see also section 5).*

may be replaced by a single one. It is also possible to build in some intelligence in the MeqParm (see fig 4).

A particular MeqSolver is normally created for a specific set of MeqParms.

6 Global Sky Model (GSM)

This is a big subject, but somewhat peripheral to this discussion. Things to be considered in the future are its relation to the Virtual Observatory (VO), and the inclusion of LSM concepts of obswin, obsres, funklets and trees (see section 5).

7 Local Sky Model (LSM)

The Local Sky Model (see fig 5) represents the subset of the GSM that is relevant to a particular observation. It contains the following main components:

- The **source list** refers to the sources that are included in the LSM. The majority will be parametrised source components (PSC), but very extended sources may be in the form of 3D images of CLEAN components.
- The **punit list** contains the *prediction-units*, i.e. groups of one or more sources that are to be predicted (for selfcal) together. A punit may be either a point-source (which is predicted by DFT), or a *patch*. The latter represents a smallish sky region that contains one or more sources (predicted by FFT, see also section 8 on uv-bricks below). Extended sources are always predicted via a patch[8].
- The **observation window** (obswin) represents the idealised shape of 'the' primary beam. It is used to make a selection from the GSM, and to speed up convergence in the determination of source parameters from residual images. If the LSM is used in 'apparent mode', the obswin has a tophat shape.
- The **observation resolution** (obsres) represents the idealised shape of 'the' point-spread-function (PSF, or synthesised beam).

Note that patches may tile the FOV just like the *facets* of residual images, but that they are not related in any way. The patch size is determined by interpolation considerations in the uv-plane, while the facet size is determined by the spatial behaviour of image-plane effects like the ionosphere. Extended bright sources will usually have a patch of their own, which may be anywhere in the sky. Punits may overlap, but a source can only be part of a single punit, of course.

8 The uv-brick concept

A uv-brick contains a 3D (u,v,f) cube of gridded uv-data. It is used (among other things) for predicting uv-model values for different ifrs, for a given LSM patch. If the patch is not too large, and the phase-centre is shifted to the centre of the patch, the resulting visibility function will be as smooth as possible, so the necessary interpoaltion of the gridded values is relatively cheap (see fig 8). *A very important feature of the uv-brick approach is that image-plane effects (which are different per ifr!) may be applied to the uv-model values by means of a few extra terms of the interpolation function.* This has been demonstrated by

[8] In the future, we may re-introduce a third category of punits, i.e. slightly extended sources with a simple shape like an elliptic gaussian. This will increase the efficiency of selfcal prediction and estimation of source parameters, but it will complicate the process of applying image-plane effects.

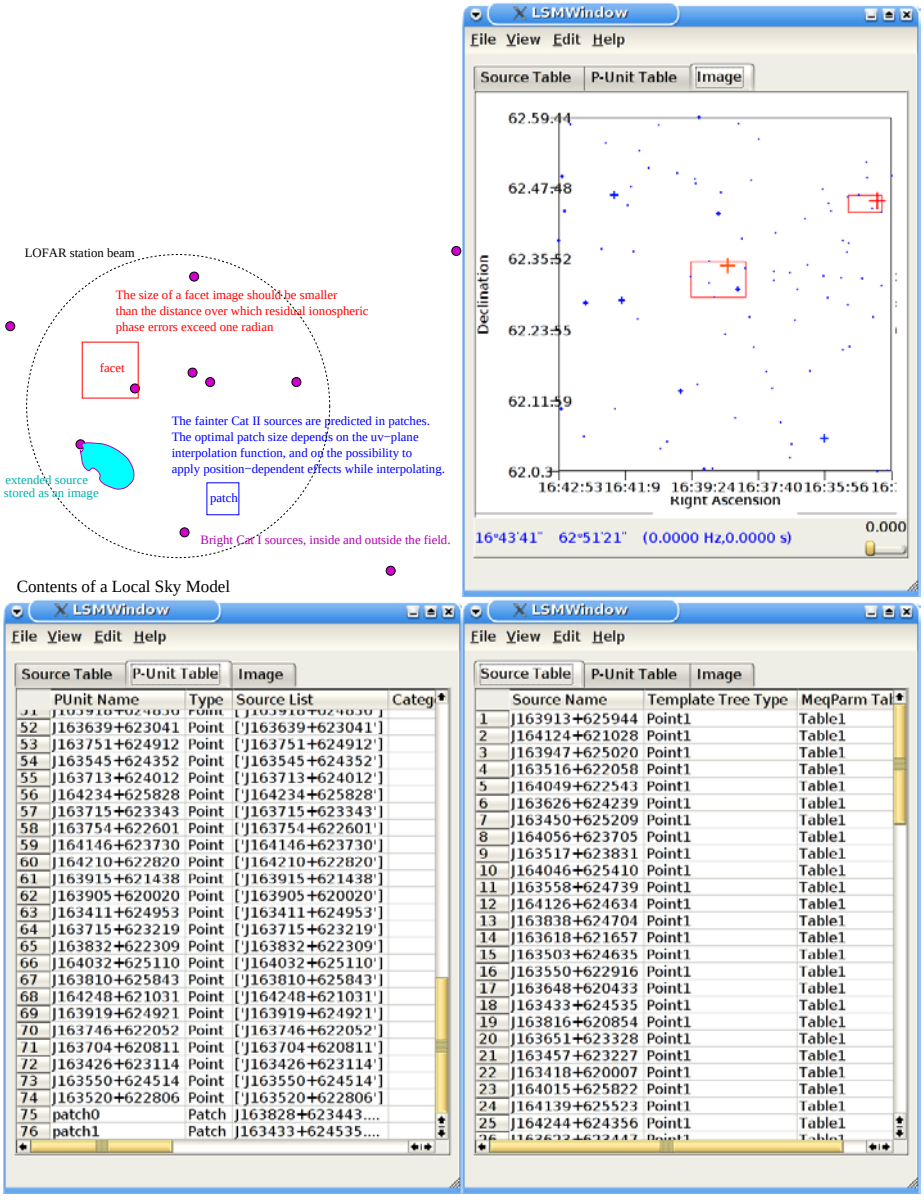


Figure 5: *Various views of the Local Sky Model (LSM). The top-left panel gives a schematic overview of its main functions. The others show the three tabs of the LSM GUI: An sky image of the sources, the punit-list and the source-list. The two dominating point sources (red crosses) are 3c343.1 (centre) and 3c343 (half-power point). The details of the two patches (red boxes) that have been defined can be inspected in the punit-list. The LSM GUI is expected to play an important role in control and monitoring of the data reduction process.*

Bhatnagar et al (cite ...) for the case of station pointing errors, and must now be generalised for other position-dependent effects like station beamshapes and ionospheric phase screens.

For efficiency reasons, the interpolation function should be as simple as possible. Therefore it will always be a 'reasonable' approximation, which will be least accurate for sources further from the patch centre. In the trade-off function between processing cost and accuracy, there will almost certainly be an optimum patch size.

Note that, since the uv-bricks are regular MeqNodes, it is perfectly possible to solve for source parameters by selfcal, just like any other M.E. parameter. However, this requires an extra uv-brick per parameter (funklet coefficient), which might be prohibitive in most cases. Still, it is nice to have the possibility in principle, and ready to use.

Essentially uv-bricks are a way of imposing large-scale constraints in the uv-plane (e.g. exploiting the smoothness of the visibility function around the phase centre). Possible other uses are redundancy calibration, or the minimising of peeling contamination. These will be discussed in a later stage.

The uv-brick concept is new, and promises to be powerful. However, since they are relatively costly in memory and processing (interpolation!), they must go through a number of optimisation iterations. But then, what is the alternative....?

9 Peeling

See fig 7.

10 Analysis of residual images

Even after subtracting a large number of Cat I and Cat II sources from the uv-data, the uv-residuals will still contain an even larger number of Cat III sources, which are too faint to be identified for inclusion in the LSM. These must be dealt with in residual images. Since it is only possible to correct uv-data for a single point in the sky, only know the PSF for that point (usually the centre of a image) will be known, and have an optimal shape for deconvolution. Towards the edge of the image, the PSF will be increasingly distorted by image-plane effects like residual ionospheric errors. Since we know these errors(t,f,u,v), it is possible in principle to calculate the PSF for any point in the image, but that is probably(?) prohibitive. It seems better to minimise the distortions by sub-dividing the FOV into smaller facets, and to make separate residual images for each.

The procedure is as follows: First the optimal facet size is estimated from the 'seriousness' of the estimated image-plane effects. Then, of each facet, the uv-data are corrected for the facet centre, and the phase centre is shifted to that position. Then the uv-data are integrated in freq and time, to limit the

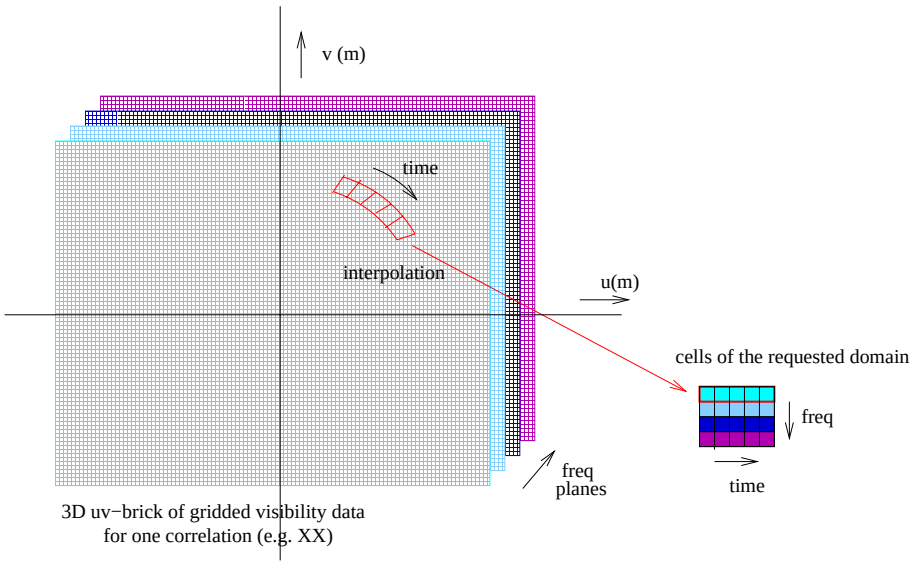


Figure 6: *A uv-brick contains a 3D (u,v,f) cube of gridded uv-data. It will be used (among other things) for predicting uv-model values for different ifrs, for a given LSM patch. The cells of a requested domain(t,f) map onto a crescent shape on the freq planes of the uv-brick, so the cell values are obtained by interpolation. This can be done relatively cheaply if the phase-centre of the measured uv-data is shifted to the centre of the patch, so that its visibility function is as smooth as possible.* ***A very important feature is that image-plane effects (which are different per ifr!) may be applied to the uv-model values as part of the interpolation function.***

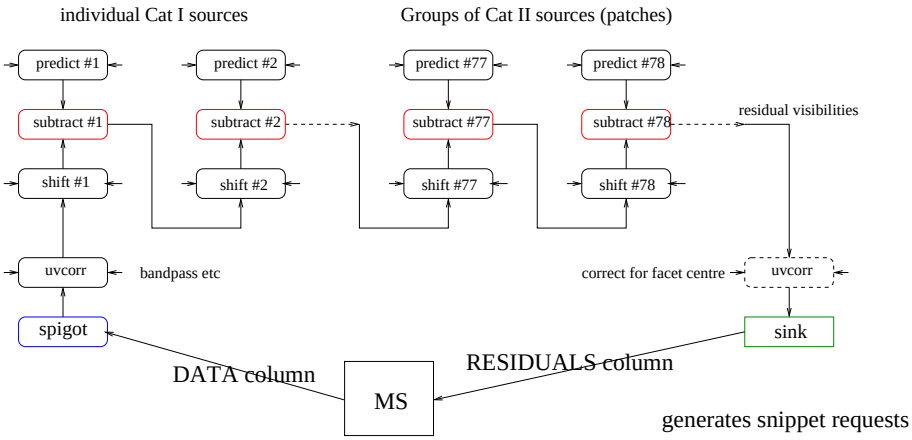


Figure 7: *Peeling is a technique in which sources/patches are treated one by one, while the phase centre is shifted to their position. This can be shown to have considerable advantages on processing efficiency. (In fact, the classic selfcal approach is probably prohibitive). For selfcal, peeling is necessarily done in order of decreasing brightness of Cat I sources. In its simplest form, it is assumed that the brightest source is the only source in the field, so one may solve for its very own M.E. parameters, and subtract it with the highest possible accuracy. Unfortunately, the selfcal solution will be influenced by 'contamination' caused by the fainter sources. Although is always possible to take these into account, this is expensive. One of the most important problems facing us is how to minimise peeling contamination as cheaply as possible. Obviously, contamination is not an issue in source/patch subtraction.*

field-of-view to the facet size. Then they are transformed a a facet residual image, to be analysed (see section 10), and stitched together into larger images again.

10.1 Update of LSM source parameters

First of all, the 4D residual images (RA, DEC, freq, stokes) should be inspected at the position where LSM sources have been subtracted. Any deviation from noise is used to updata the parameters of that source in the LSM. This is done by solving for these parameters, using its the MeqTree in the LSM that defines the relation between its parameters and its manifestation (I,Q,U,V) in the image. Thus, it is a task for the LSM object (see section 5).

Note that the source residuals may not only be caused by inaccurate source parameters, but also by residual instrumental errors. Obviously, the LSM is not able to solve for those. They will disappear when going through the entire selfcal loop again, using the updated source parameters.

10.2 Dealing with Cat III sources

As more and more data is accumulated (some LOFAR observations will integrate on the same field for hundreds of hours), the noise level in the residual images will decrease. This makes it possible to detect more and more Cat III sources, and include them into the LSM (thereby making them Cat II sources). This implies that they get their own MeqTree, of course. For compact sources, it is probably sufficient to assume that they are point sources, and use their peak value in the image as a first estimate of its parameters.

What remains are the extended Cat III sources, which will often be the sources we are most interested in. They will require som sort of deconvolution with a variable PSF...

10.3 Some novel imaging artefacts

The specific hardware implementation of LOFAR, and its calibration system, will introduce imaging artefacts of a type that we are not used to in our present telescopes. Some examples:

- Remnants of Cat I subtraction due to peeling contamination
- Remnants of RFI
- BSR effect (a large-radius halo around bright sources?)
- Pattern of facet edges (Cat III imaging)
- Sidelobe confusion (PSF sidelobe level)
- Pattern of LSM patch edges (Cat II subtraction)
- Etc....

Some of these effects have been analysed to some extent (cite BSR), but it remains unclear what their precise impact on the image will be in practice. In addition, there are bound to be other ones, that we have not yet thought of. In some cases, the artefact may be impossible to get rid of entirely, so the best we can do is make sure that it cannot be confused with a scientifically interesting feature like the EoR. In all cases, the only way ahead is to use adequate tools to patiently chip away at the problem.

11 Ionosphere (MIM)

Over the years, the ideas for an ionospheric model for LOFAR have undergone a considerable evolution. It has been clear for a long time that the 'Zernicke' method used by Bill Cotton for VLA 74 MHz reduction only really works for arrays smaller than 10 km. It can also be demonstrated that open-loop methods (like using a 3D model, or extrapolating from other observations as frequencies) are inherently too inaccurate. Thus we started out with independent phase-screens for each station (and each beam), which required up to 20 calibrator sources in the field that had to be bright enough to give SNR>3 in 10 s. Remarkably, there turned out to be enough such sources to make this work at most LOFAR frequencies, but the total number of parameters was clearly prohibitive. More careful thought produced ever more economical models, until it was realised that, if we may only assume that the ionosphere is smooth and large-scale, it is possible to define a Minimum Ionospheric Model (MIM) with a surprisingly small number of parameters: as few as 20 under good conditions, valid for the entire sky, and the entire frequency range. As conditions get worse, the number of MIM parameters is (automatically!) increased, until some reasonable limit is reached. For more details, see the separate MIM document (cite ....).

However, to answer the critcism of Colin Lonsdale at the recent SKA WFI meeting, it is only fair to note that these MIM parameters describe the large-scale phase function that is interpolated to subtract the fainter sources. They are derived from the N selfcal phases in the direction of one or more bright calibrator sources, in which N is the number of stations. Since these bright sources will be subtracted with their own phases for greater accuracy, these should strictly be added to the total nr of MIM parameters.

The MIM approach has recently been tested in simulations by James Anderson (JIVE), and the results are encouraging. However, since MIM sounds too good to be true, it will remain a little controversial until it has really been demonstrated in the field.

Finally, there might be more areas where a MIM approach can offer a (very) substantial reduction in the number of parameters in the LOFAR M.E. Since this is by no means a luxury, it should be vigorously pursued. Essentially, we should look at areas where we are not interested in the internal structure of a disturbance, but only in its phaenomenological effect on the observations.

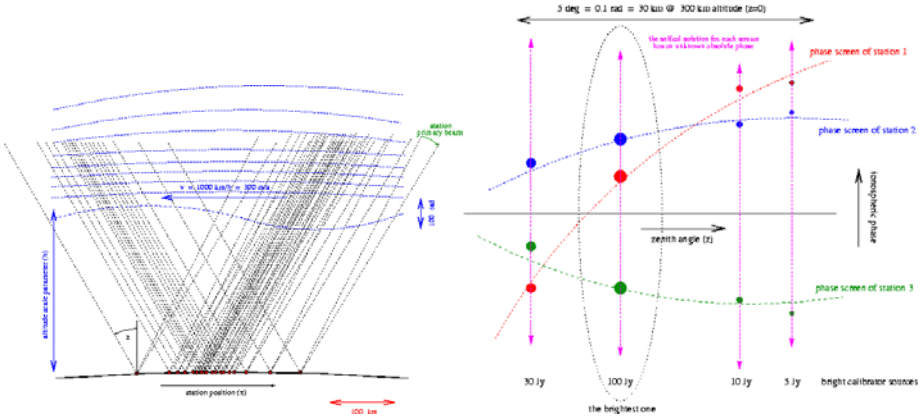


Figure 8: *The Minimum Ionosperic Model (MIM) only deals with the effect on the data, i.e. it totally ignores the physics or any internal structure of the ionospheric blanket. MIM just assumes that the ionosphere is smooth in all dimensions (az,el,x,y,freq,time), and postulates a smooth phase function, e.g. a low-order polynomial. Its parameters ae estimated from selfcal on a bright calibrator source in the field. If the ionosphere is relatively quiet, MIM will have fewer parameters than the number of LOFAR stations, so one calibrator will be sufficient. This situation can be detected by checking whether the selfcal phases (red dots) and the MIM phase screen (dashed red line) actually coincide. If that is not the case, more terms should be added to the MIM function. The selfcal solution of other calibrator sources (blue, green) should also be consitent with the corresponding MIM function. See section 11 for more details.*

12 Station beam shapes

The LOFAR station primary beams will have much higher sidelobes than the parabolic dishes that we are used to. In the words of Jaap Bregman: "LOFAR imaging is all-sky imaging". This means that we will have to calculate and subtract the effects of many more sources in the sidelobes than the usual 'A-team' (Cas A, Cygnus A, ...) and the Sun. This task is complicated by the fact that the beamshapes, and certainly their side-lobe patterns, will be different for different stations. Moreover, because the response patterns rotate w.r.t. the sky, outlying sources will rapidly move from sidelobe to sidelobe. Fortunately, their effect in the image will be attenuated considerably by freq and time smearing effects. **In any case, it will be firmly assumed here that closed-loop techniques will be required to measure the station voltage beams continuously.** Examples of open-loop schemes are calculating beamshapes from dipole gains, or attempts to make sure that all station beams have the same shape. These may certainly helpful to create the best possible starting conditions, but they will not be sufficient by themselves[9].

The brightest sources in the sidelobes will simply be treated as regular Cat I sources, i.e. we will solve for the instrumental parameters in their direction. Those baselines for which their contribution sinks (temporarily) below the noise will be automatically ignored in the solution. Because they are sufficiently bright, and accurately known, we will somehow contrive to deal with sources crossing sidelobe boundaries, and do so asynchronously.

However, there will almost certainly also be a class of fainter Cat II sources in the sidelobes, i.e. sources that are not bright enough (or too difficult to model) to be used for selfcal, but that still cause image artefacts, or increase the noise. In the main lobe, Cat II sources are subtracted by interpolating its known shape(s). But, as stated above, it will be assumed that we do not know the shape of the sidelobes, but only the gain in the direction of a few Cat I sources. *Obviously, there is an unsolved problem here.* We should decide whether it is better to have:

- a larger main lobe and smaller sidelobes
- station sidelobes that are so different that they cancel each other in ifrs, as much as possible.

Another feature of the horizontal LOFAR dipoles is a much larger instrumental polarisation, wihich increases with zenith distance. This is because the difference between the two voltage beams per station. We can solve for the voltage beam shapes, and the absolute ionosphere(?), but we have very few polarised calibrators for Faraday calibration.

Voltage beams are complex. It would be nice if we could get away with assuming that the phase is caused by the ionsphere, while the beamshape response is purely real. This might work in the main lobe (?), but certainly not in the

[9] If this statement is wrong, all the better. But it seems unwise to assume it.

side-lobes, since beam phases do not cancel (the phase changes by π between sidelobes).

Summarising, the LOFAR station voltage beam shapes are will present us with a host of interesting new problems. A simulation tool has been developed with MeqTrees (see fig 9) to help analyse them, and to help make decisions about station antenna configurations. The resulting MeqTrees can be directly grafted into the LOFAR M.E.

In the meantime, calibration continues to require that the beamshape parameters are smooth in all dimensions. This means that HBA beamformer coefficients are allowed to jump every 10 minutes, but not more often. Adaptive beamforming is discouraged, but should be done in such a way that the coefficients are changed 'sufficiently' smoothly.

13 Simulations

The MeqTree system has most of the necessary features to play an important role in simulations for LOFAR, SKA, etc. The following things have to be developed:

- The GSM needs to be filled, with real and simulated sources of all kinds.
- Ideally, the M.E. used for corrupting the data with simulated instrumental errors should be different, but equivalent, to the one used for calibration.
- Tools must be developed to compare input and output in a meaningful way.

At this moment, according to its chairman, the SKA Simulation Working Group does not have a suitable tool to do its work. We could contribute via the SKADS effort, lead by Mike Garret.

14 ITS calibration

Cotton Zernike will work for station (MIM?). Use Cas A and Cygnus A. Horizon calibrators can be included in LSM with RA,DEC(time).

15 Conclusions

LOFAR calibration will not be easy. It might take several years before we really get to the observation noise. On the positive side, there is enough information (bright sources) to solve the problem in principle, and we have a lot of new ideas to try. In addition, the MeqTree system is a powerful development tool, which promises a low threshold for the participation by bright people around the globe. The stage is set for the development of the LOFAR Measurement Equation.

Problems that need special attention in the near future:

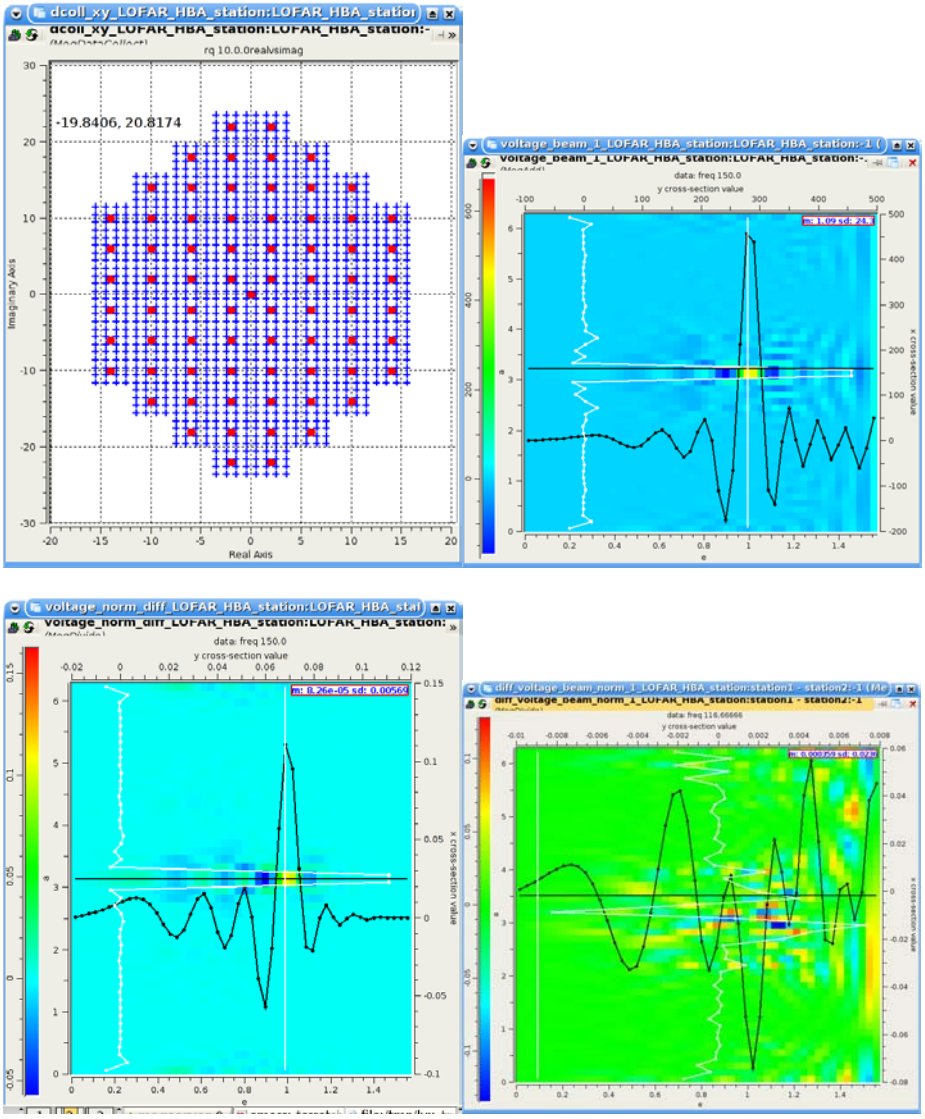


Figure 9: *Simulation of LOFAR HBA station beams, using MeqTrees. The top-left (meqbrowser) image shows the HBA antenna configuration, while top-right shows one of the two voltage beams @ 150 MHz, when pointed at azimuth (a) of π rad, and elevation of 1.0 rad. Bottom-left shows the normalised difference between two voltage beams, and bottom-right shows the normalised difference between corresponding ('X') voltage beams of two identical LOFAR HBA stations, where the one is rotated 0.2 rad w.r.t. the other. It is difficult to say in this stage what these figures mean exactly for LOFAR calibration, but it is clear that such simulations will be very useful in getting (long overdue) answers to questions like that.*

- Cat II sources in station beam sidelobes
- MIM validation
- Minimisation of peeling contamination
- Application of image-plane effects to extended sources
- etc

Targeted programs will be started in the coming months.

References

[1] Peeling the Visibility Onion (JEN 200)

[2] Description of MeqTree system (OMS)

[3] Description of PSS5 (JEN, 2004)

[4] Local Sky Model (LSM) functional decsription (JEN, RJN, 2005)

[5] uvbrick implementation (RJN, 2005)

[6] Applying image-plane effects (Bhatnagar, Cornwell and Golap, 2003)

[7] MeqParm functional description (JEN, MXM 2005)

[8] A Minimum Ionosphere Model (JEN 2005)

[9] Verification of Minimum Ionospheric Model (MIM) (Anderson, JEN, 2005)

[10] Thoughts on the Global Sky Model (GSM) (OMS, 2002)

[11] Description of the Bandpass Sawtooth Ripple (BSR) effect (JEN, 2003)

[12] ADASS posters (various, SBY, MXM, RJN, OMS, JEN, 2002-5)

[13] Analysis of the peeling algorithm (Jeffs, van der Tol, 2005)

[14] Operation 343 (Brentjens, 2005)

[15] Using peeling in Miriad (Oosterloo, 2005)

[16] The MeqTree module (JEN, URSI New Delhi 2005)

[17] LOFAR Calibration (JEN, SPIE Glasgow 2004)

[18] Modelling the LAR primary beam (Willis, 2005)

A Working together

Two things seem to be clear. The first is that we absolutely need to work together on a global scale to meet the challenges of calibrating the new generation of giant telescopes. The second is that the astronomical software community has not been very good in distributed development[10]. The latter is no reason for despair, as long as we heed the lessons from the less successful projects[11], and try to understand why others have blossomed.

Our role models are highly successful packages like MatLab, Python and Linux. They have all managed to get their users to produce shareable modules around a robust kernel. The nevessary conditions are that the kernel (and the contribited modules) offer functionality that users really want, that the kernel interfaces are simple and stable, and that the kernel is perceived to be 'fast enough'.

The composition of the MeqTree team reflects the strategy of involving a wide range of people in the development, and of exercising the software on real data, by people that are actively interested in the results. The present team members are from ASTRON/TL (Noordam, Smirnov, Nijboer, Brentjens), ASTRON/RO (Mevius, Oosterloo, Assendorp), the LOFAR EOR key project (Yatawatta), DRAO Penticton (Willis), TU Delft (van der Tol). It can also draw on active consultants (Brouw). As the entry threshold is lowered, the group will be gradually extended with others with interesting skills or problems, who can work in the kind of environment we can offer in this stage. They may be located anywhere in the world, and work with data from various existing telescops (e.g. GMRT, ATCA, VLA74MHz).

The execution of the exercise program in section 2 will take several months. After each successful stage, the MeqTree system should be more interesting for experienced outsiders who asolutely need the more complex Measurement Equations offered by MeqTrees. Conscious efforts are made to make the threshold as low as possible for them, by providing a system of helpful TDL scripts for inclusion, cannibalisation, demonstration and testing. But ultimately, they will have to to the specifics themselves, and thereby become valuable members of the MeqTree core community. They will also prove highly valuable as a buffer between regular users and the core development team. The first example of this 'elite inner circle' is Tony Willis, who uses MeqTrees to simulate the CLAR

[10] In fact, the optimum appears to be a team with a maximum of 4 highly talented maniacs, who happen to tolerate each other, located in a dark corridor of the same building, zealously shielded from formal project management. Running an astronomical software project has been described as an attempt in 'herding cats'.

[11] It is important to note that AIPS++ has been very successful in several areas. It has produced a number of modules (tables, measures, fitting, imaging) that are widely used in various contexts. (It should be noted that these modules are all well-defined, and each has been produced by a single person). It contains a superset of all known imaging algorithms, and is still actively devloped in this area. It has been adopted as the official package for ALMA, which must be good for speed and stability. And, as part of its inscrutable process, it has produced the Measurement Equation. Its most important failure was that it was never exercised properly, mainly because the development team did not agonise about making that attractive for working astronomers.

primary beam.

B Interfaces with other parties

B.1 Interface with key science projects

EOR, Transients, Pulsars, Surveys

B.2 Requirements imposed by calibration

Smoothness in all dimensions (t,f,l,m). Ionosphere. Bandpass shape. Pointing and tracking (jump every 10 min reluctan tly allowed)

B.3 Consultation with the LOFAR BBS development team

C MeqTrees

Even though this document is about the LOFAR calibration strategy, and not about the implementation of the working prototype, here is a quick reminder of the basic features of the MeqTree module.

C.1 Tree Definition Language (TDL)

The Tree Definition Language (TDL), developed by Oleg Smirnov) is used to generate MeqTrees by means of Python scripts. In combination with the MeqTree browser (see fig 11) it is fast and convenient, and offers a very short turnaround time for experimentation[12]. Here is a very simple example of a TDL script:

```
ns = nodescope()

p1 = ns.parm1 <\,{}< Meq.Parm(array({[}1,0,0{]},{[}0,0,0{]}))

c1 = ns.const1 <\,{}< Meq.Constant(complex(-1,3))

root = ns.sum <\,{}< Meq.Add(p1, c1)
```

Of course there is a lot more to it, but this is the essence. A MeqNode of a particular type is defined by Meq.Type(), with a list of is zero or more children as arguments. These may be followed by some named arguments, which specify values for specific fields in the state record of specific node types. The argument of a leaf node like a MeqParm is converted into a default funklet.

[12] Since the introduction of TDL, a huge amount of (fairly powerful) Glish code for generating trees has been dropped overnight.

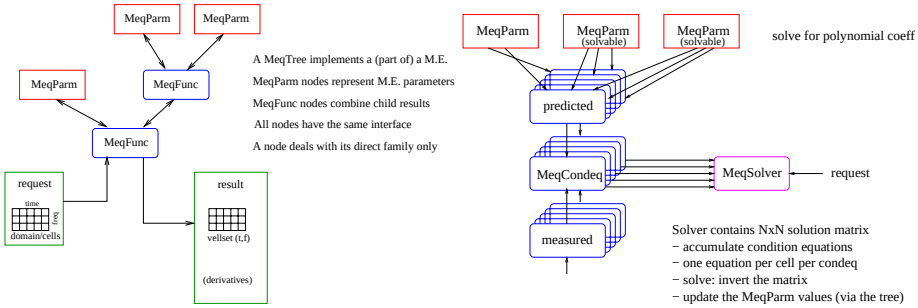


Figure 10: *MeqTrees are used in the working prototype to implement an arbitrary M.E., and to solve for (arbitrary subsets of) its parameters. A tree (a graph, really) consists of nodes (objects), which are strung together by means of a minimal interface. Each node has an internal state (including a result cache), and has zero or more children. When a node receives a request to calculate a 'vellset' of values for the cells of a given domain, it passes this request to its children, and waits for their results. Depending upon its type, it then combines these into a result of its own (e.g. a MeqAdd node adds the results of its children). A node with zero children is a 'leaf', and represents an M.E. parameter (MeqParm), a constant or a data 'spigot'. It has access to its own resources (e.g. an MS or a MeqParm table) to satisfy a request. Each node has a result cache, which can be used to avoid re-calculation when the same request is received from different parents. A 'smart caching' scheme seeks to optimise the balance of processing vs memory use.*
The panel on the right illustrates a simple solver tree. The children of a MeqSolver node are MeqCondeq nodes, which construct condition equations (one per cell) for the solution matrix. They do this by calculating the difference between its two children, which represent measured and predicted data values. They are also able to calculate derivatives w.r.t. each solvable, i.e. the the funklet coefficients of those upstream MeqParms that are set to 'solvable'. After collecting all equations, the MeqSolver inverts the matrix, and passes the incremental improvements back up the tree to the relevant MeqParms.

C.2 The MeqBrowser GUI

The MeqBrowser GUI (see fig 11) is a powerful tool for the generation, control, and inspection of MeqTrees. It has the following main features:

- Script editor. Once a TDL script is load, it may be edited. The result is saved automatically when executing it. Any python errors are clearly indicated, even in imported scripts (which are then loaded automatically, and can also be edited).
- Tree browser shows the structure of the tree, i.e. the relation between parents and children. In addition, the contents of all nodes may be inspected individually. Nodes may be inspected by class, or from the root node(s).
- Tabbed grids of display panels, to display the contents of nodes in various ways. The bookmarks menu allows rapid access to built-in views of multiple nodes that belong together.
- Forest state record. Contains overall control parameters and other useful information. It can be inspected by clicking on it in the browser section, just like the contents of a node.
- Forest save/restore. Once a forest of connected node objects has been generated, it may be saved/restored as a binary file. This includes any cache results, which is useful for bug reporting.
- Request editor. A tree is executed by offering a request (see fig 10) to one of its nodes, usually a root node (i.e. a node without parents). The request (domain, nr of cells) may be edited.
- Stream control. If a forest is connected to an MS (using MeqSpigot and MeqSink nodes per ifr), execution may be controlled by means a special GUI, which specified MS name, freq channels, integration time, etc.
- Publishing results. Each node may be told to broadcast (publish) every new result it produces. This can be picked up, e.g. by a display tool.
- MeqParm fiddler. Any node can serve as the starting point for a special GUI, which gives access to all the MeqParms upstream. These may be inspected and edited, after which the tree may be re-executed. This is a truly wonderful debugging feature.
- Debugger functionality. Execution may be halted, interactively or by means of breakpoints at selected nodes. There are various options for step-by-step execution.
- Profiler. Every node keeps a record of its execution statistics. These may be collected by the profiler (debug menu), and displayed per node class. This is very useful for identifying processing bottlenecks.

It cannot be stressed enough that the functionality of the MeqBrowser GUI enormously reduces the turnaround time for experiments and debugging. If anything, this speeds up development time proportionally, and reduces the chance of undetected errors.

C.3 Lowering the threshold

MG scripts (examples, demonstration, experimentation, importable functions) and 'official' TDL scripts.

D Glossary

Since the LOFAR calibration system necessarily contains a few innovations, there is a need for some (carefully chosen) new terms.

- domain/cells/snippet: A domain is a rectangular area in N-dim space, usually freq-time. In a request, a domain is sub-divided into cells, for which values are required. A snippet is a smallish domain, e.g. in the order of minute in time.
- funklet/polc: An N-dimensional array of coefficients of smooth functions, e.g. polynomials. Different functions are possible in different dimensions. M.E. parameter (MeqParm) values are in the form of funklets, and the coefficients are the unknowns that are being solved for. A polc is a special kind of funklet, with only polynomials, usually freq-time.
- imaging: Transforming uv-data into an image (i.e. we do not subscribe to the custom of including selfcal and deconvolution in imaging)
- leaf (node): A node without children, e.g. a MeqParm or a MeqConstant or a MeqSpigot (data interface). A leaf has access to its own information to satisfy a request for values.
- (meq)node: A C++ object that can be connected to other nodes to make a MeqTree.
- patch/facet: A patch is a smallish area on the sky, which is used by the LSM for source prediction. A facet is a sub-division of the FOV, used to reduce the effect of imaging errors by image-plane effects.
- request/result/vellset: A request contains a domain, subdivided in cells. When passed to a node, the latter returns result that contains a vellset with values for the requested cells. Requests and results may also contain other information.
- residual uv-data/image: Because of image-plane effects, all known sources should be subtracted from the uv-data. The resulting residual uv-data are then transformed into (facet) residual images, for further analysis.

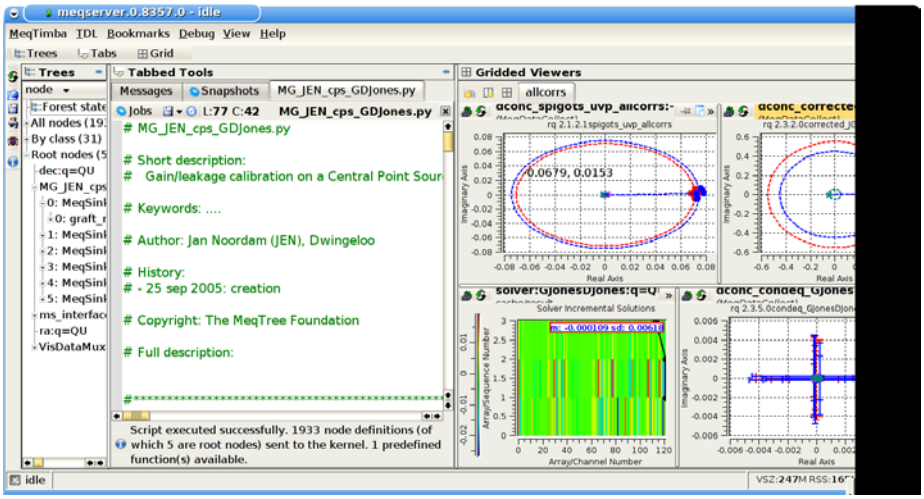


Figure 11: *The MeqBrowser GUI is a powerful tool for the generation, control, and inspection of MeqTrees. When a TDL (python) script is loaded, it is displayed in the central section, where it may also be edited(!). MeqTree nodes are generated by pushing the blue button above the script, which executes the _define_forest() function in the script. The tree is executed by selecting one of the functions in the 'jobs' menu, which are also in the script. This causes request(s) to be passed to suitable root node(s) of the tree.*
The structure of the tree(s) can then inspected in the browser section on the left. The statue of any node may be inspected by clicking on it, which causes its contents to be displayed in the display section on the right. Many TDL scripts define in-built views of one or more related nodes that are accessible by the bookmarks menu at the top. The example above shows such a view of the uv-data before and after solving simultaneously for complex gain (GJones) and leakage (DJones) of 4 WSRT telescopes, using real data (XX=red, YY=blue, XY=green, YX=magenta). The model is a polarised point source at the field centre. The bottom panels show the MeqSolver metrics for 3 iterations, and the MeqCondeq residuals.
The MeqBrowser has many more useful features, which are listed in section ... They are described in more detail elsewhere (cite ...). But the best way is to just play with it, which, very importantly, is fun.

- solver/condeq: A solver collects condition equations from its condeq children, and accumulates them into a solution matrix. It inverts te matrix, and feeds the incremental results back up the tree to the (solvable) MeqParms, whose values are then updated. Condeq nodes construct condition equations from the difference between their two children (which represent measured and predicted values), and dervatives w.r.t. the (coeff of the) solvable MeqParms.

- spigot/sink/s2s: Data interface nodes. A MeqSpigot reads uv-data for a particular ifr from a column of the MS, at full (f,t) resolution. A MeqSink writes it back into a column. During data-reduction, requests are generated by the MeqSinks. An s2s tree represents a single ifr, from spigot to sink.

- tree/forest: A MeqTree has a single root node. A group MeqTrees is called a MeqForest

- Timba: Derived from 'timber'. Unofficial alternative name for MeqTree.

....to be completed....