




## SDP MEMO 97: DIRECTION DEPENDENT SELF CALIBRATION IN ARL

Document Number..... SKA-TEL-SDP-000097  
 Document Type..... REP  
 Revision..... 01  
 Author..... T.J.  
 Cornwell  
 Date..... 2019-1-14  
 Document Classification..... Unrestricted  
 Status..... Released

Name	Designation	Affiliation	Signature	
Authorized by:				
Tim J. Cornwell	Lead Consultant	TCC		
			Date:	15/1/2019
Owned by:				
			Date:	
Approved by:				
			Date:	
Released by:				

Paul Alexander	SDP Project Lead	University of Cambridge	<i>Paul Alexander</i> <small>Paul Alexander (Jan 15, 2019)</small>	
			Date:	

## DOCUMENT HISTORY

Revision	Date Of Issue	Engineering Change Number	Comments
01	2019-01-15		Prepared for SDP CDR review

## DOCUMENT SOFTWARE

	Package	Version	Filename
Word processor	Google Docs		SKA-TEL-SKO-0000000-01_GenDocTemplate
Block diagrams			
Google docs Add-ons	<a href="#">Cross Reference</a>		Used for figure & table numbering and references.
	<a href="#">Table of contents</a>		Used for heading numbering.
	<a href="#">List of figures</a>		Used to generate list of figures and tables

## ORGANISATION DETAILS

Name	SDP Consortium
Lead Organisation	The Chancellor, Masters and Scholars of the University of Cambridge The Old Schools Trinity Lane Cambridge CB1 1TN United Kingdom
Website	<a href="http://www.ska-sdp.org">www.ska-sdp.org</a>

© Copyright 2019 University of Cambridge



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

## TABLE OF CONTENTS

### [1 Introduction](#)

### [2 Scope](#)

### [3 Overview of existing direction dependent self-calibration algorithms](#)

#### [3.1 The need for direction dependent calibration](#)

#### [3.2 Existing algorithms](#)

### [4 Model Partition Calibration Workflow](#)

### [5 Implementation in ARL](#)

#### [E-step](#)

#### [M-step](#)

### [6 LOW Simulations](#)

#### [Station layout:](#)

#### [Sky model:](#)

#### [Ionospheric model:](#)

#### [Types of simulation:](#)

#### [Construction of the initial skymodel:](#)

#### [Summary of results:](#)

### [7 Commentary](#)

## LIST OF ABBREVIATIONS

<b>API</b>	Application Programming Interface
<b>ARL</b>	Algorithm Reference Library
<b>COTS</b>	Commercial off the shelf
<b>EF</b>	Execution Framework
<b>SDP</b>	Science Data Processor
<b>SIP</b>	SKA Integration Prototype
<b>SKA</b>	Square Kilometre Array

# 1 Introduction

Correction of direction dependent effects (DD) when imaging the sky, especially at low frequencies, is an important driver of SDP capabilities. The algorithms are complex and the processing loads are usually very high (See e.g. [RD01], [RD03], [RD04], [RD05], [RD06], [RD07], [RD08], [RD13]). The SDP platform will have to run very demanding pipelines for the relevant science cases, especially for LOW where the ionosphere is particularly troublesome and the calibration requirements (for EOR) are very stringent.

Hence we need to understand whether the SDP Processing Architecture can support direction dependent calibration. The architecture has not been demonstrated at SKA scale, even for simple workflows, so any answer to this question must currently be provisional and of the same nature as the scaling of simpler processing such as imaging with time-variable direction independent (DI) errors. With this in mind, we have developed a proof of concept demonstration of DD calibration. This demonstration is similar to other DD calibration algorithms, and we do not claim any substantial algorithmic advances. The intention in this memo is to describe how DD calibration can be implemented in the SDP processing architecture as exemplified by the ARL.

## 2 Scope

The scope is to develop and demonstrate an algorithm for direction dependent self-calibration in ARL, building on and augmenting the current capabilities. The question is whether the SDP processing architecture, as exemplified in ARL, can support DDE self-calibration.

The demonstration should include:

- Realistic direction dependent effects, such as will be expected for the ionosphere when observing with LOW,
- Realistic sky model,
- Distributed processing, using Dask,
- Consensus optimisation

## 3 Overview of existing direction dependent self-calibration algorithms

### 3.1 The need for direction dependent calibration

The visibility  $V$  measured by two antennas ( $i, j$ ) separated by vector  $b$  (in wavelengths) for a brightness distribution  $I$  in direction  $s$  is given by the following equation:

$$V_{i,j}(\underline{b}, t, f) = \int E_i(\underline{s}, t, f) E_j^*(\underline{s}, t, f) I(\underline{s}) e^{2\pi j \underline{s} \cdot \underline{b}} d\underline{s}$$

If the direction dependent terms  $E$  are constant or all equal and the field of view is sufficiently small, this can be approximated by a two dimensional Fourier Transform. Thus, for example, a fixed antenna primary beam response can be absorbed into the sky brightness  $I$ .

The common examples of non-constant  $E$  patterns are:

- Primary beams vary with frequency.
- Asymmetrical antenna primary beams that rotates on the sky with parallactic angle. MID antennas are designed to have very low asymmetries. In principle, these can be measured and corrected.
- Station beam for an aperture array such as LOW as seen from different aspects. These are dependent on a number of factors, including station layout, direction-dependent coupling effects, polarisation, and signal path variations..
- Atmospheric phase variations due to troposphere or ionosphere. The ionosphere is particularly problematic for LOW because of the rapid time and angle variations at the frequencies at which LOW operates.

If the  $E$  terms are not Direction Dependent (DD) then the calibration problem amounts to determining the sky brightness and one  $E$  term for each dish/station as a function of time and frequency. This Direction Independent (DI) calibration problem, called self-calibration, is amenable to least squares solutions for the  $E$  terms.

### 3.2 Existing algorithms

There are a number of established algorithms for direction dependent calibration. These vary in intent and in computing requirements.

### 3.2.1 Peeling

Peeling deal with bright sources seen through parts of the primary beam that are known with insufficient accuracy. The  $E$ 's are variable and essentially unknown in the direction of a bright source, with the consequence that error patterns are spread over the entire field. In peeling, a component is placed on a known source location, the gain is solved for that component and removed from the visibility. This can be performed sequentially and/or iteratively. After all such sources have been estimated the main field can be imaged and self-calibrated.

### 3.2.2 Field calibration

At low resolution, the main effect of the ionosphere is to shift sources. For a field of bright sources, the shift across the field can be estimated and a predictive model fit to e.g. a set of Zernike polynomials. The use of a screen model of this sort lowers the dimensionality of the fitting process, thus increasing stability and robustness. This model works well for baselines less than about 5-10 km.

### 3.2.3 SPAM

SPAM [RD05] is similar in approach to Field Calibration but an explicit phase screen at the nominal height of the ionosphere is fit.

### 3.2.4 SageCAL

In SageCAL [RD01], the target signal, the Epoch Of Reionisation (EOR) signature, is assumed to be low level and little affected by calibration errors, but it is obscured by bright foreground sources for which calibration errors are important. Thus the bright foreground sources are simply to be removed, with no correction of the background signal. Many sources are estimated and removed, in clusters to decrease the computational load. Once all sources are removed the EOR signal should be revealed.

### 3.2.5 Facet Calibration

In Facet Calibration [RD07], the field of view is tiled into facets around the brightest sources. Each facet has its own phase behavior. Each facet image can be estimated by self-calibration once the contributions of all other facets have been removed. Thus all fields are iterated, gradually decoupling the imaging and self-calibration. At the end, the result is CLEAN image and associated calibration table, one per facet. The final image can be constructed by combining the facets.

## 4 Model Partition Calibration Workflow

As seen in the previous section, there are a number of existing algorithms for imaging in the presence of direction dependent effects. We have developed a framework that can serve as a basis to implement such algorithms within the SDP processing architecture. This framework draws upon the algorithms described in section 3.

The full calibration and imaging problem can be conceptualised as a large problem in non-linear least squares minimisation, where the unknowns are the gain solutions and the sky model parameters. This is computationally infeasible. Typically, therefore, decoupled parts are solved alternating between each. The ICAL pipeline is one step down this path but for non-isoplanatic imaging more flexibility is required. The Model Partition Calibration algorithm (see e.g. [RD01]) works by dividing the calibration and imaging problem into different partitions. A model partition consists of a number of related model parameters, such as the time series of the ionospheric phase for a particular direction on the sky. A key feature of these partitions is that solution for the different partitions is loosely coupled. The modelled visibilities are the sum of the partition visibilities.

A partition typically has two types of parameters: the first type describes part of the sky, and the second type describes the calibration associated with that part of the sky. The sky-model part of a component may be, for instance, a single source, a group of sources, an image describing diffuse emission, or a facet. The calibration associated may be, for instance the gain solutions of the antennas.

An important degenerate case is a single sky model and a single calibration solution, observed with multiple frequencies, that are distributed by frequencies. This is ICAL. The coupling between the gain solutions across the different frequencies could be included directly in the solver, but it also could be enforced by the consensus operation. The simplest case would be to take the average (or a robust statistic) across the different frequencies. Although this is not optimum for SNR, it probably is sufficiently good for many cases, especially given the iterative nature of the algorithm.

The MPC algorithm is shown below in pseudo-code.

---

**Algorithm:** MPC Consensus Optimisation Algorithm

**Approach:** We wish to model an observed visibility set by a set of parameters  $\underline{\theta}$ . In principle, the parameters can be solved by a nonlinear least squares (NLSQ) fit. However this scales as the square of the length or worse of  $\underline{\theta}$ . Instead we partition the model into  $P$  sets  $\underline{\theta}_p$ , and solve each subset separately, having first removed



the estimated visibility from all other partitions. This must be iterated O(10)-O(100) times but the overall cost in CPU and working set is much lower than for the NLSQ approach.

**Goal:** Estimate model parameters for mixture of Visibilities:

$$V = \sum_p V_p(\underline{\theta}_p)$$

**Variables:** Observed visibility  $V_{obs}$ , partitions of model parameters  $\underline{\theta}_p$ , partition visibilities  $V_p$

**Outputs:** partition parameters  $\underline{\theta}_p$ ,  $P$  partition visibilities  $V_p$

**Steps:**

1. Initialise:  $\underline{\theta}_p, V_p$
2. Iterate  $n$  steps or until convergence:
  - 2a. Iterate over all model partitions  $p$ :
 

E-Step:

$$V_p^{(n+1)}(\underline{\theta}_p^{(n)}) = V_{obs} - \sum_{q \neq p} V_q^{(n)}(\underline{\theta}_q^{(n)})$$

M-step:

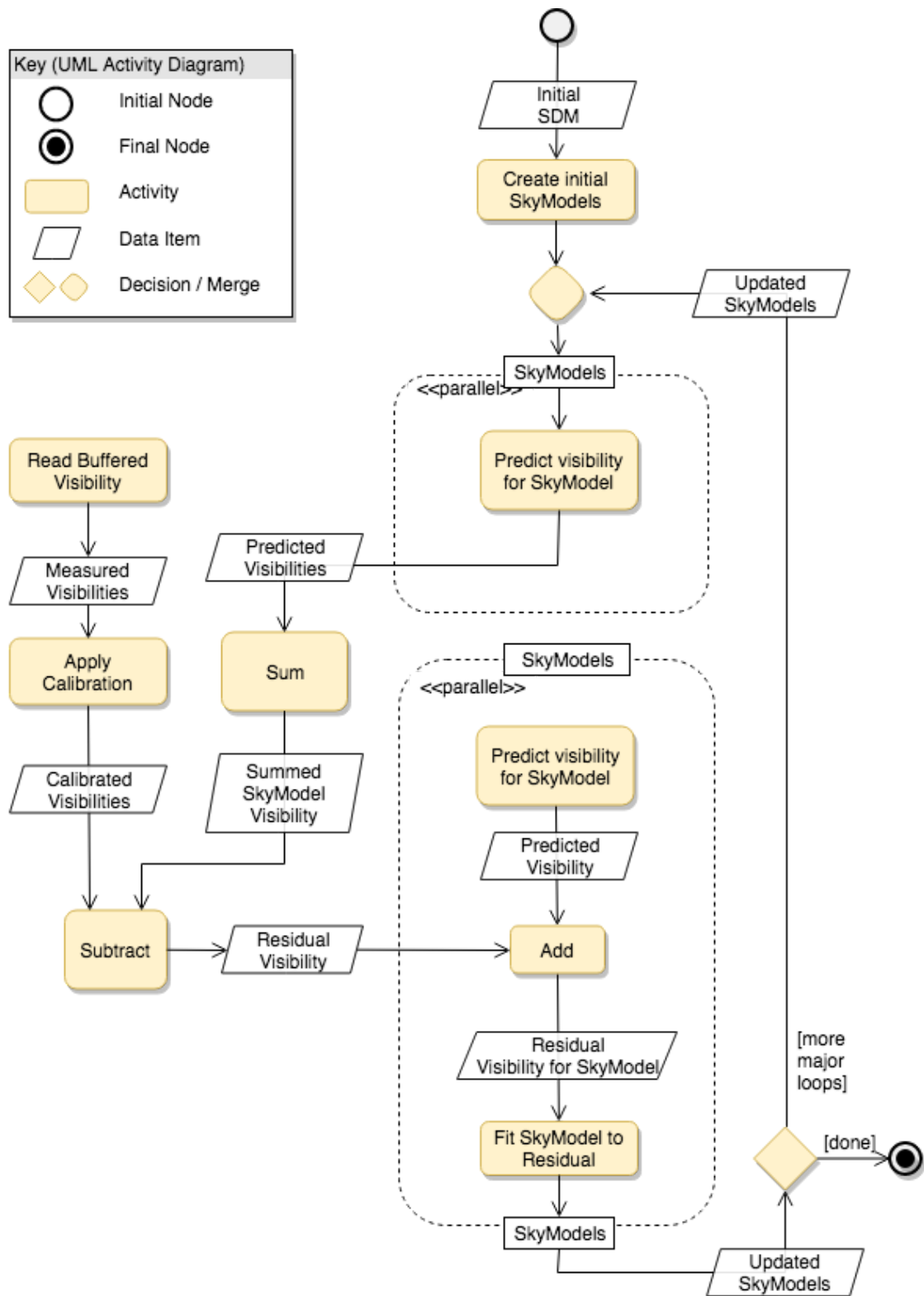
$$\underline{\theta}_p^{(n+1)} = \operatorname{argmax}(f(V_p^{(n)}(\underline{\theta}_p^{(n)})))$$
  - 2b. Consensus:
 
$$\underline{\theta}^{(n+1)} = CO(\underline{\theta}^{(n+1)})$$

---

**Algorithm 1:** Model Partition Calibration CO algorithm

This algorithm, perhaps apart from the consensus operation, is actually a simple and robust approach, which has no doubt been used previously. It is an iterative divide-and-conquer approach: guess the separate partitions in some way, correct with the observed data, and then use some approach to improve the parameters from the estimate of the partitioned data. It is similar in concept to the major/min cycle deconvolution algorithm and the self calibration approaches.

There are some aspects of this algorithm that are potentially concerning. For example, it is not at all clear that there are unique solutions. However, those concerns are not important here.



**Figure 1:** Activity diagram for Model partition algorithm. The Model partition algorithm is modeled on SageCAL, Peeling. The MPC has the goal of finding a set of SkyModel's that reproduce the observed data.

## 5 Implementation in ARL

We now describe the implementation in ARL and execution via Dask.

As a reminder, the core processing capabilities in the SDP processing architecture are located in the processing components module. These can be invoked from workflows, which are run via an execution framework.

The functions below named *\*\_list\_arlexecute\_workflow* are all workflows, written using the *arlexecute* interface. The *arlexecute* interface is a thin layer that allows execution of the same workflow using Dask, Daliuge, or serially. If other versions were desired, such as MPI, corresponding versions name *\*\_list\_mpi\_workflow* would be required, and could be derived from the *arlexecute* versions.

The presence of *list* indicates that the function operates on a list of items (such as skymodels), and/or may return a list of items (such as images).

The combination of (skycomponents, image, mask, and gaintable) is a single skymodel.

The structure of a SkyModel is shown below.

**Table 1:** Structure of SkyModel

Data member	ARL Type	Comments
image	Image	Image to be processed by FFT's.
mask	Image	Mask of valid pixels in the image e.g. a patch
components	List[SkyComponent]	Compact components to be processed via direct Fourier transform
gaintable	GainTable	Applies to all data predictions

Given such a skymodel, we can predict the visibility due to that partition/skymodel. The resulting visibility is called the datamodel. At convergence the sum of the data models should be close to the observed visibility.

$$V = \sum_p V_p(\theta_p)$$

The goal of MPC is to find a set of skymodels that are consistent with the observed visibility i.e. the predicted visibility matches the observed visibility. This is done by alternating between two steps: the Expectation and the Maximisation Steps.

The Expectation step isolates an estimate of each datamodel by subtracting all other data models.

### E-step

Visibility predictions  $V_p(\theta_p)$  are calculated from the skymodel parameters  $\theta_p$  using *predict\_skymodel\_list\_arlexecute\_workflow*. The steps in the function are:

- Iterate over all skymodels
  - Process component part of skymodel
    - DFT components to visibility
  - Process image part of sky model
    - Multiply skymodel image by mask,
    - Predict using the usual *predict\_list\_arlexecute\_workflow* or *predict\_list\_serial\_workflow*, that predicts a list of visibility sets from a list of images,
  - Sum image and component parts
  - Apply skymodel gaintable to the predicted visibility,
  - Add to a list of the datamodels.

At any one iteration, each data model is updated using the residual visibility for that skymodel. The function *extract\_datamodel\_list\_arlexecute\_workflow* performs this update:

$$V_p^{(n+1)}(\theta_p^{(n)}) = V_{obs} - \sum_{q \neq p} V_q^{(n)}(\theta_q^{(n)})$$

That concludes the E-step. Next we follow the Maximisation Step which refines the skymodels based on these updated datamodels.

### M-step

Once the data model decomposition is available, we update the skymodel by any of a number of approaches. We write these as the results of a maximisation step.

$$\underline{\theta}^{(n+1)} = \text{argmax}(f(\underline{\theta}^{(n)}, V(\underline{\theta}^{(n+1)})))$$

Both the image and gaintable parts of the skymodel  $\underline{\theta}_p$  must be updated. To update the image part of each skymodel we perform deconvolution of the dirty image for each skymodel. The function *invert\_skymodel\_list\_arlexecute\_workflow* calculates the dirty images, essentially the reverse of the predict step:

- Iterate over all data models and sky models
  - Correct data model for skymodel gaintable
  - Invert corrected data model using the usual *invert\_list\_arlexecute\_workflow* or serial version.

The dirty images can be deconvolved using e.g. MSCLEAN along with a mask, and the images in the skymodel updated appropriately. However, since we already have a model which can be used as the starting point of the deconvolution. We use a function *convolve\_skymodel\_list\_arlexecute\_workflow* to calculate the dirty image that would be obtained for the skymodels, without any calibration correction. The steps are:

- Iterate over sky models
  - Predict visibility data for each skymodel using *invert\_list\_arlexecute\_workflow*
  - Invert predicted data using the usual *invert\_list\_arlexecute\_workflow* or serial version to obtain the dirty image for each skymodel.
  - Subtract from the dirty images made using gaintable corrections, thus forming a residual image for each skymodel.

This allows the deconvolution to be differential. The residual images are then processed using *deconvolve\_list\_arlexecute\_workflow*, which allows a range of algorithms and scalings.

To update the gaintables in the skymodels, we use least squares fitting of the datamodels to the predicted visibility, as is normal in calibration, using the function *calibrate\_list\_arlexecute\_workflow*.

In our current processing model without MPC, different visibility sets (e.g. frequency ranges) are already distributed across different nodes. In MPC, we have to calculate multiple copies of the same visibility sets, one per skymodel. So the storage and calculation problem has increased by the number of skymodels. In our MPC work, Dask itself will move data as required.

## 6 LOW Simulations

In order to be able to validate the algorithm, we simulated data from LOW observations at a single frequency of 100MHz. The observation was 600s of 10s integrations, around transit. The field centre declination was -40 degree. Since the w-term is not the focus of this investigation and the total observing time is very short, we set the w coordinate to zero.

### Station layout:

The processing scales strongly with number of stations. To reduce the number of stations we only used  $\frac{1}{3}$  of stations in the core, and  $\frac{1}{6}$  outside the core. This configuration has 92 stations. See figure 2 for the configuration.

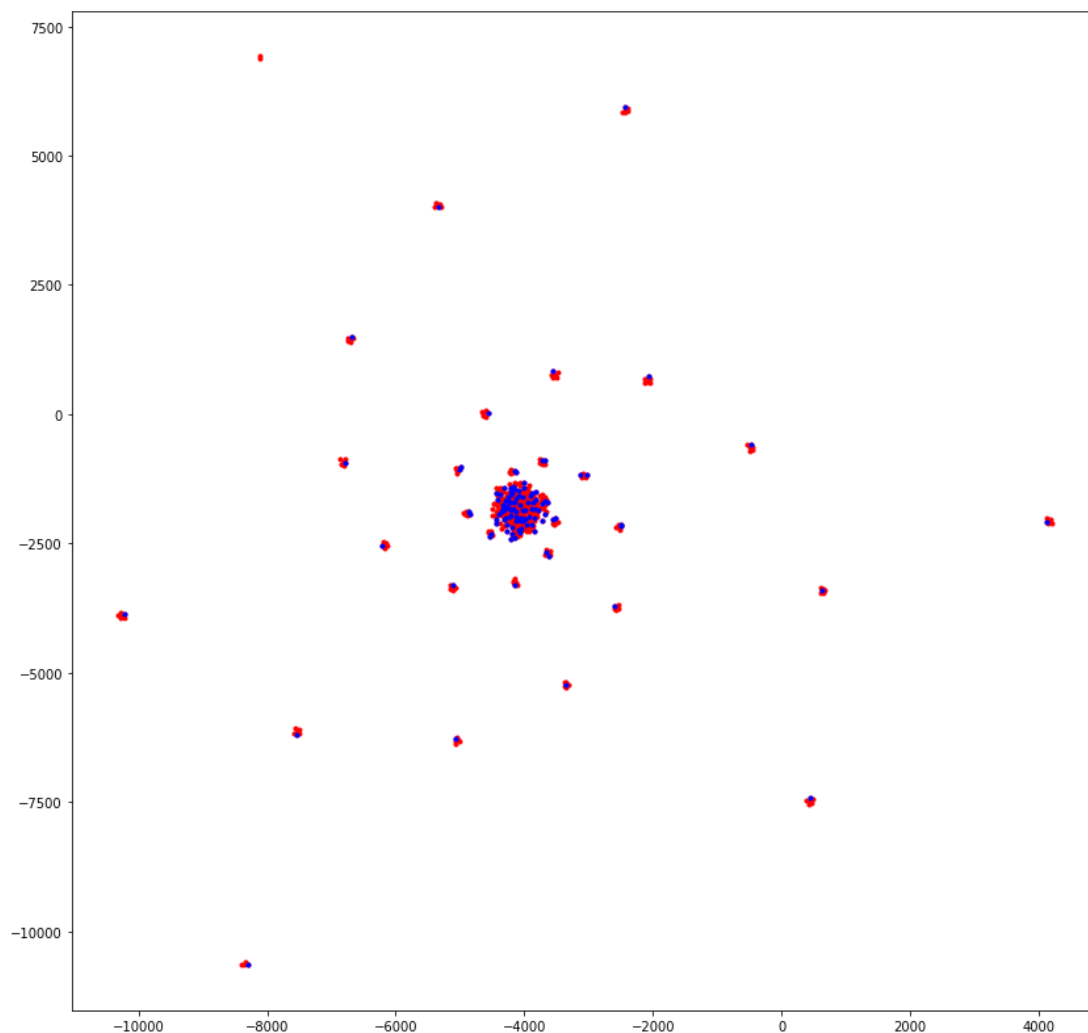


Figure 2: Stations using in simulations. Red are the originals in LOW, blue are those selected.

## Sky model:

We constructed a sky model from a set of components generated from GLEAM, and applied a primary beam constructed using OSKAR. See Figure 3.

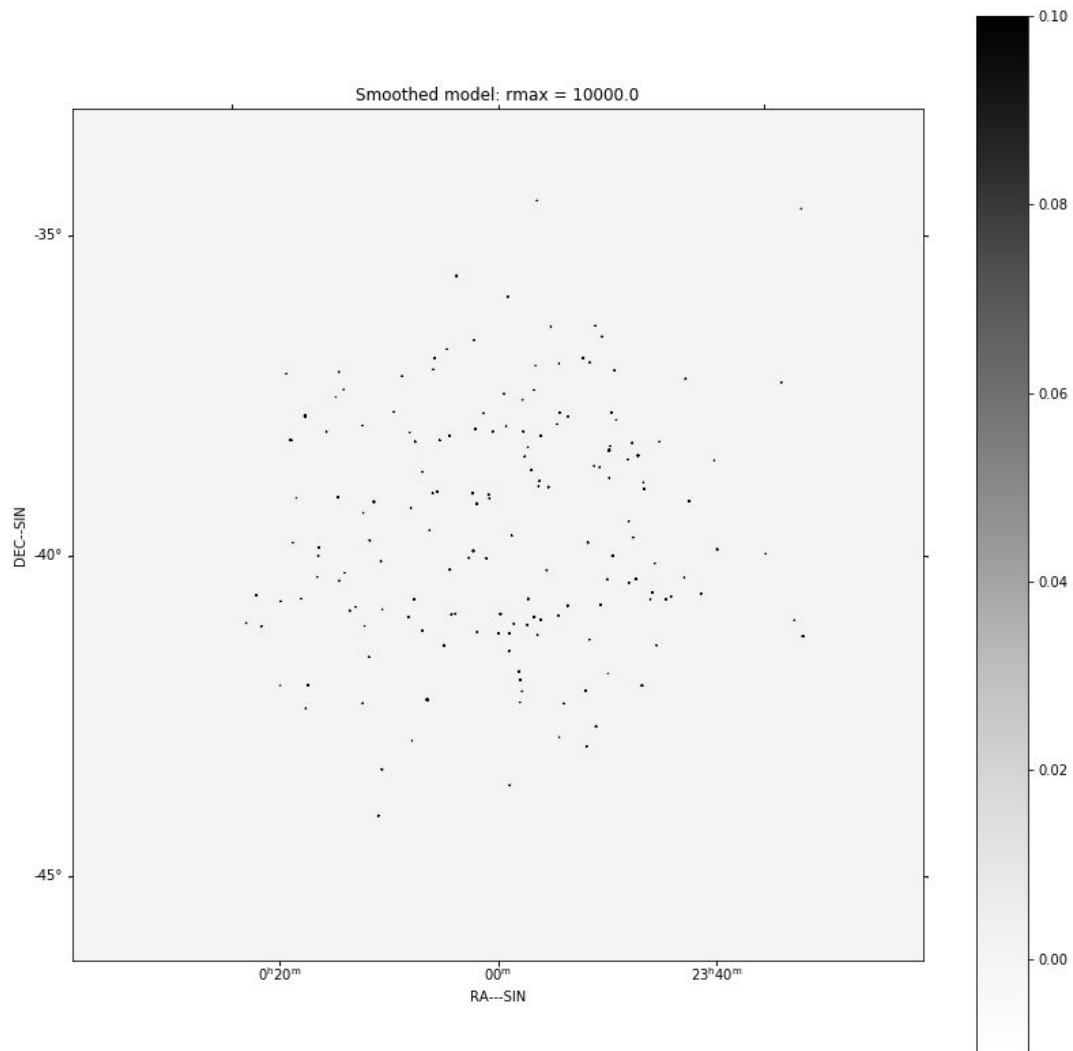
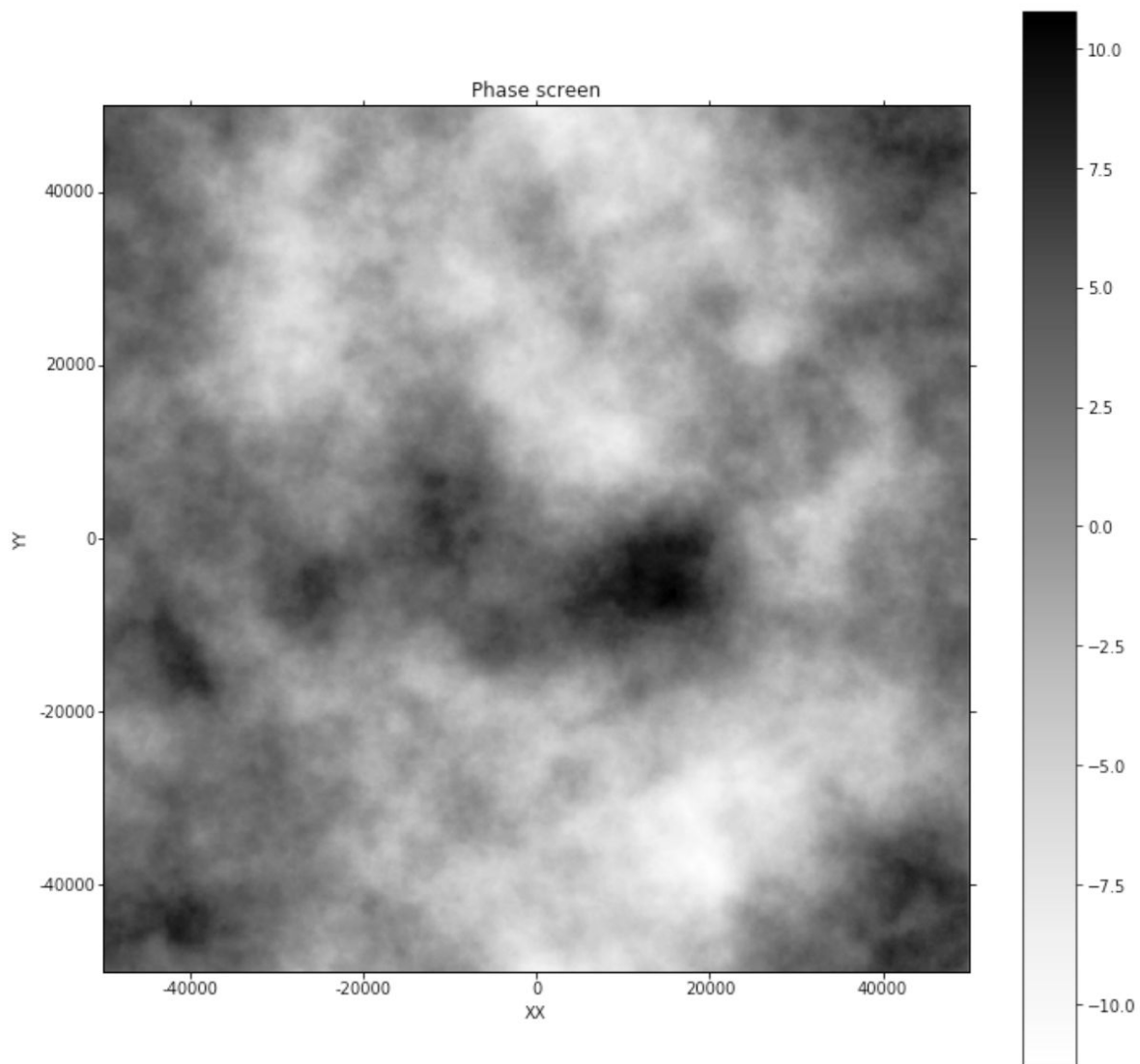


Figure 3: Smoothed model, formed from GLEAM catalog, selecting all sources  $> 0.2\text{Jy}$ . The brightest source in the field is  $4.88\text{Jy}$ . A fixed primary beam, calculated using OSKAR has been applied.

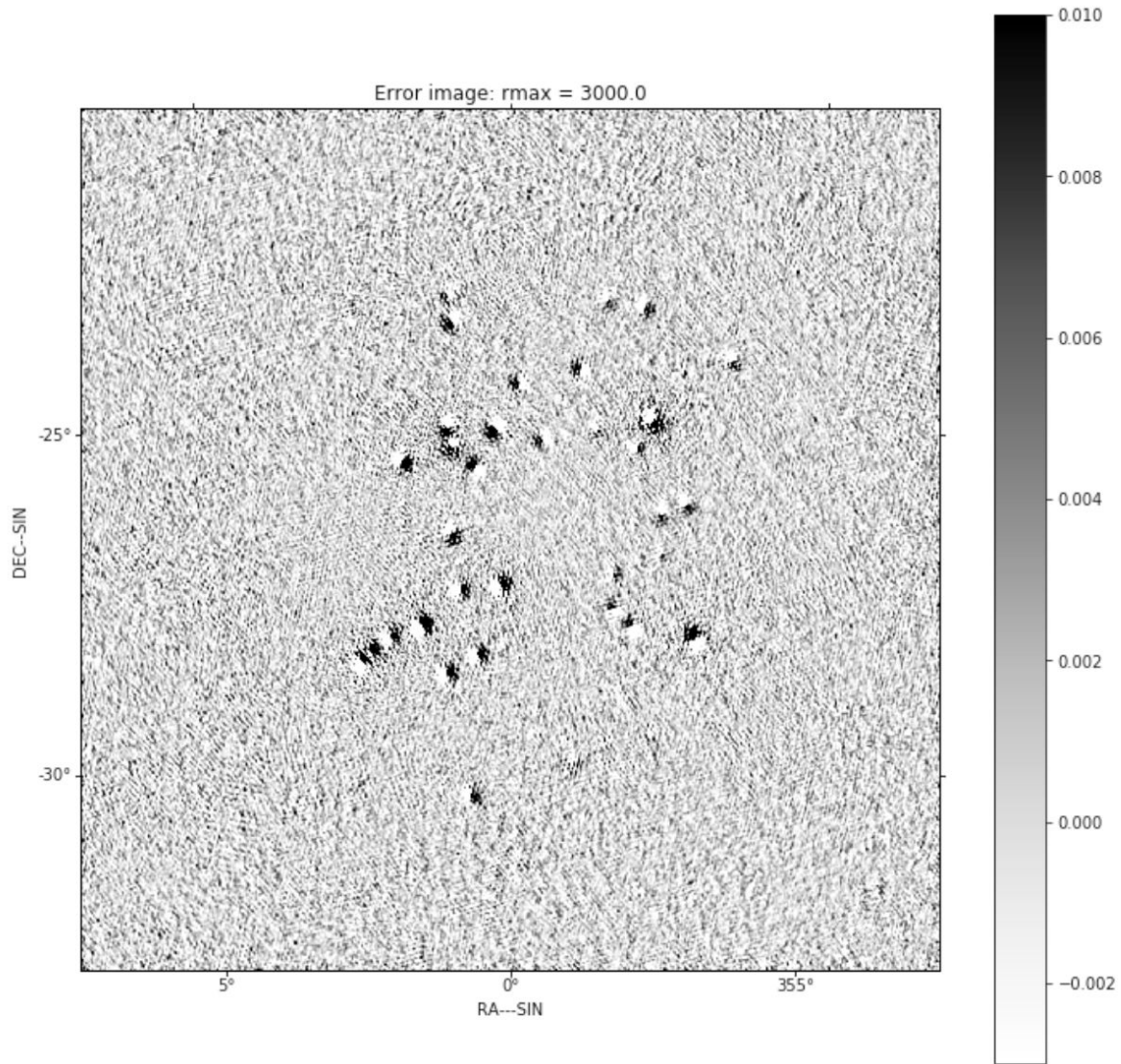
## Ionospheric model:

The ionospheric model was based on the work and python code of Srinath et al. [RD09, RD10]. In order to limit the processing costs, we used the characteristic scale  $r_0 = 5\text{ km}$ . See figure 4 for an example of the phase screen. This level of phase error causes source shifts on  $3\text{ km}$  baselines (see figure 5) and significant non-isoplanatism on  $10\text{ km}$  baselines (see figure 6).

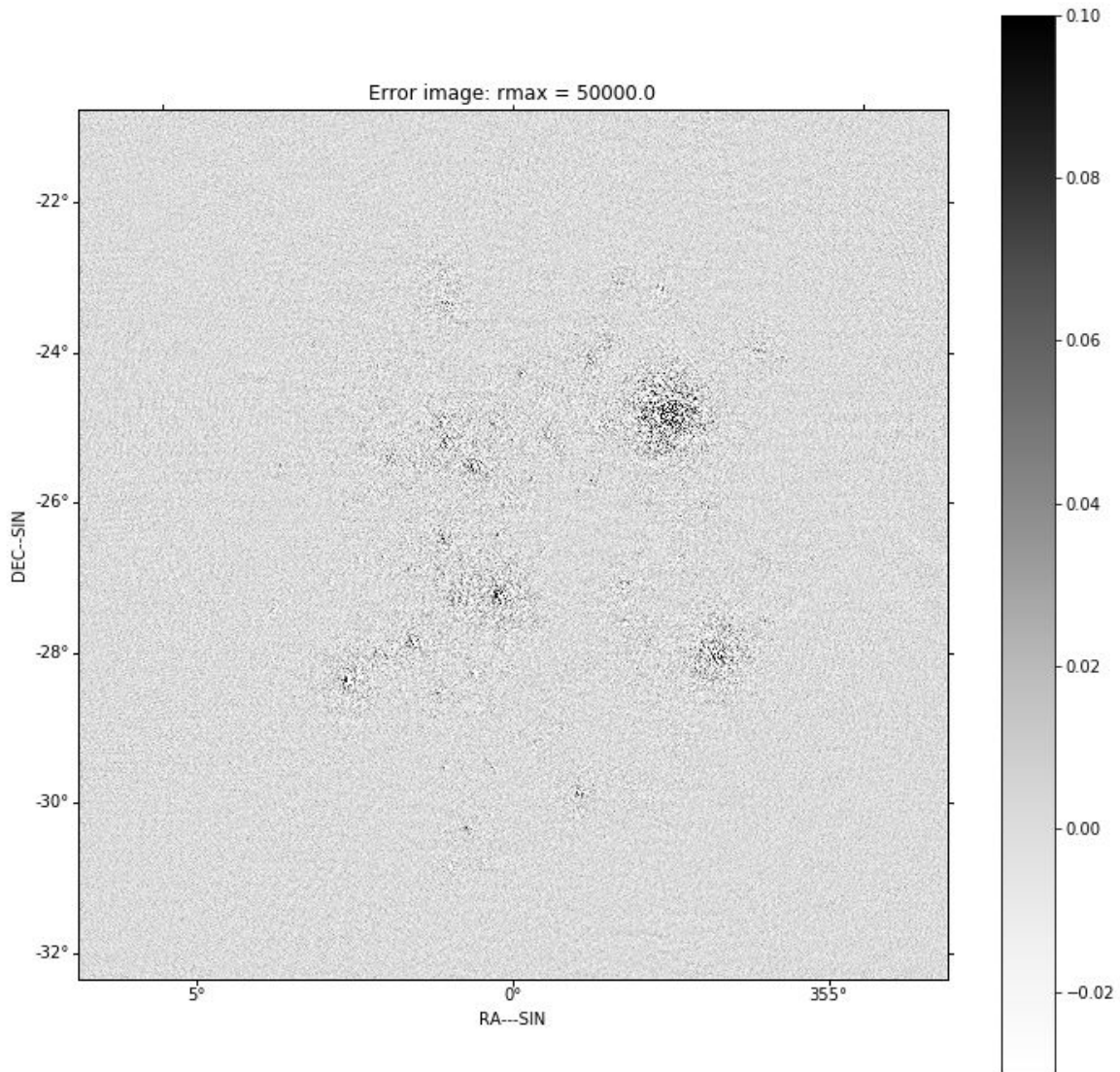


**Figure 4:** Simulation of a phase screen at 300km height using the Python code of Srikar Srinath [RD10]. The screen is a single layer moving westwards at 150 km/s. This was captured in a FITS file with third axis the observing time 600s in 10 second intervals. This corresponds to a Kolmogorov spectrum with index  $-5/3$  compared to typical values of 1.7 - 1.8. In order to limit the processing costs, we used the characteristic scale to be  $r_0=5\text{km}$ . This causes significant non-isoplanatism on 10km baselines.





**Figure 5:** Error image for GLEAM image maximum baseline 3km. The peak errors are manifestly non-isoplanatic and mostly source shifts, as expected.



**Figure 6:** Error image for LOW GLEAM image, maximum baseline 80km, at 100MHz. The peak errors are manifestly non-isoplanatic and are no longer simple source shifts.

The ionospheric model also suggests a possible mechanism for consensus optimisation. We can project the gain phases obtained back onto the ionospheric screen and then read back a new set of gaintable values, such as just the average phase in each phase screen cell.

### Types of simulation:

Two different data sets were simulated:

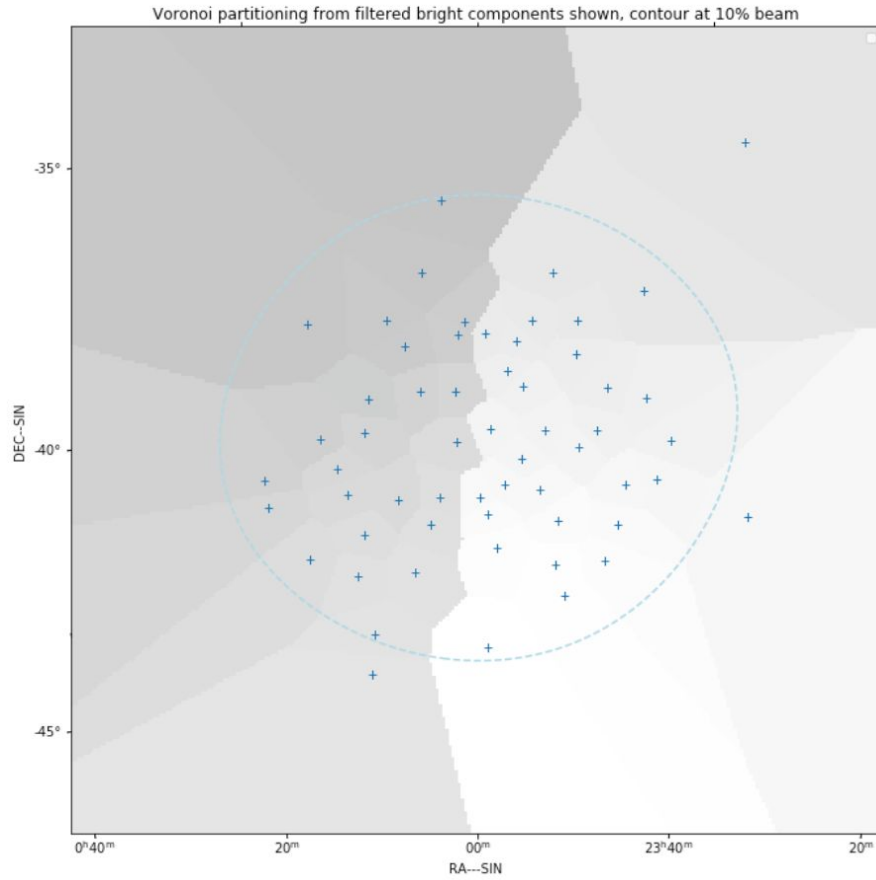
- **ISO:** A single set of time-variable and station-dependent phase errors were applied to each component. **ISO** should be amenable to direction independent calibration.
- **NONISO:** The phase of the visibility phase for each component was calculated from the difference in phase at the two different ionospheric pierce points. By construction, **NONISO** is not amenable to direction independent calibration.

## Construction of the initial skymodel:

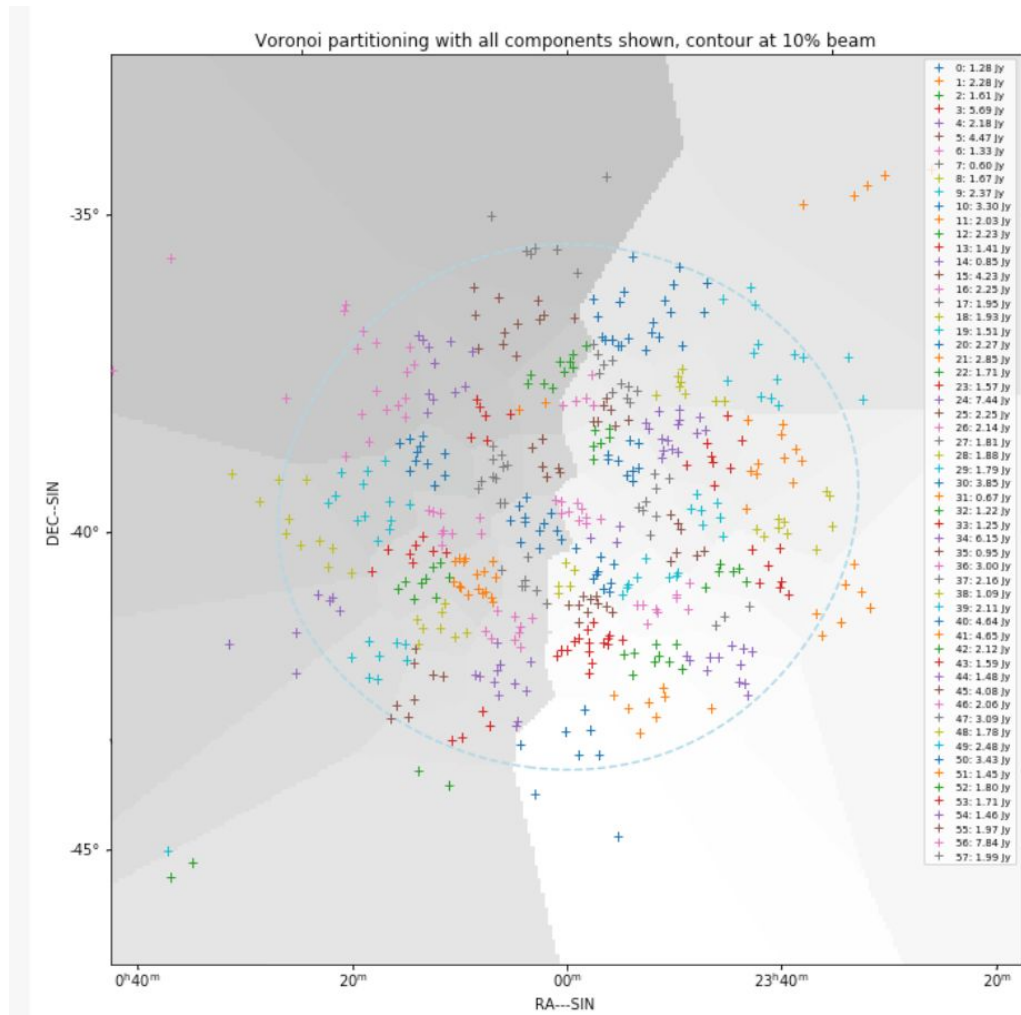
It is necessary to partition the skymodels such that a single skymodel roughly covers an isoplanatic patch. A single skymodel is partitioned by:

- Identification of the brightest components found using isoplanatic calibration and imaging.
- Down-selection to the brightest of two neighbours separated by less than a minimum allowed distance (e.g. 0.01 radian)
- Assignment of one skymodel per bright component, and the component direction of the component as the phase centre of the gaintable.
- Construction of image mask for each skymodel. The mask is the set of pixels for which have the same nearest bright component.
- Assignment of skycomponents to the nearest bright skycomponent.
- Construction of a list of skymodels, one per bright component.

An example of partitioning is given in Figures 15 and 16. In this case the components are drawn from the GLEAM catalog, the primary beam is applied, and bright components selected to be brighter than 0.3Jy. In Figure 16, we show the assignment of weak components to the Voronoi regions. In practice, we do not use the weak components in each skymodel but instead represent the sky with masked images.



**Figure 15:** Partitioning of a set of components drawn with flux  $> 0.05$  Jy from the GLEAM catalog. The primary beam is applied and then the bright components are selected to be all  $> 0.3$  Jy. The gray scale outlines the Voronoi regions. Sources closer than 0.01 rad have been downselected to the brightest. This leaves 58 of the original 111 sources to define the separate skymodels.



**Figure 16:** Partitioning of a set of components drawn with flux  $> 0.05$  Jy from the GLEAM catalog. The primary beam is applied and then the bright components are selected to be all  $> 0.3$  Jy. The gray scale outlines the Voronoi regions. Sources closer than 0.01 rad have been downselected to the brightest. This leaves 58 of the original 111 sources to define the separate skymodels. The aggregate fluxes for each partition are shown in the legend. Although we show all components, the actual skymodel only contains the bright components that defined the partitioning.

### Summary of results:

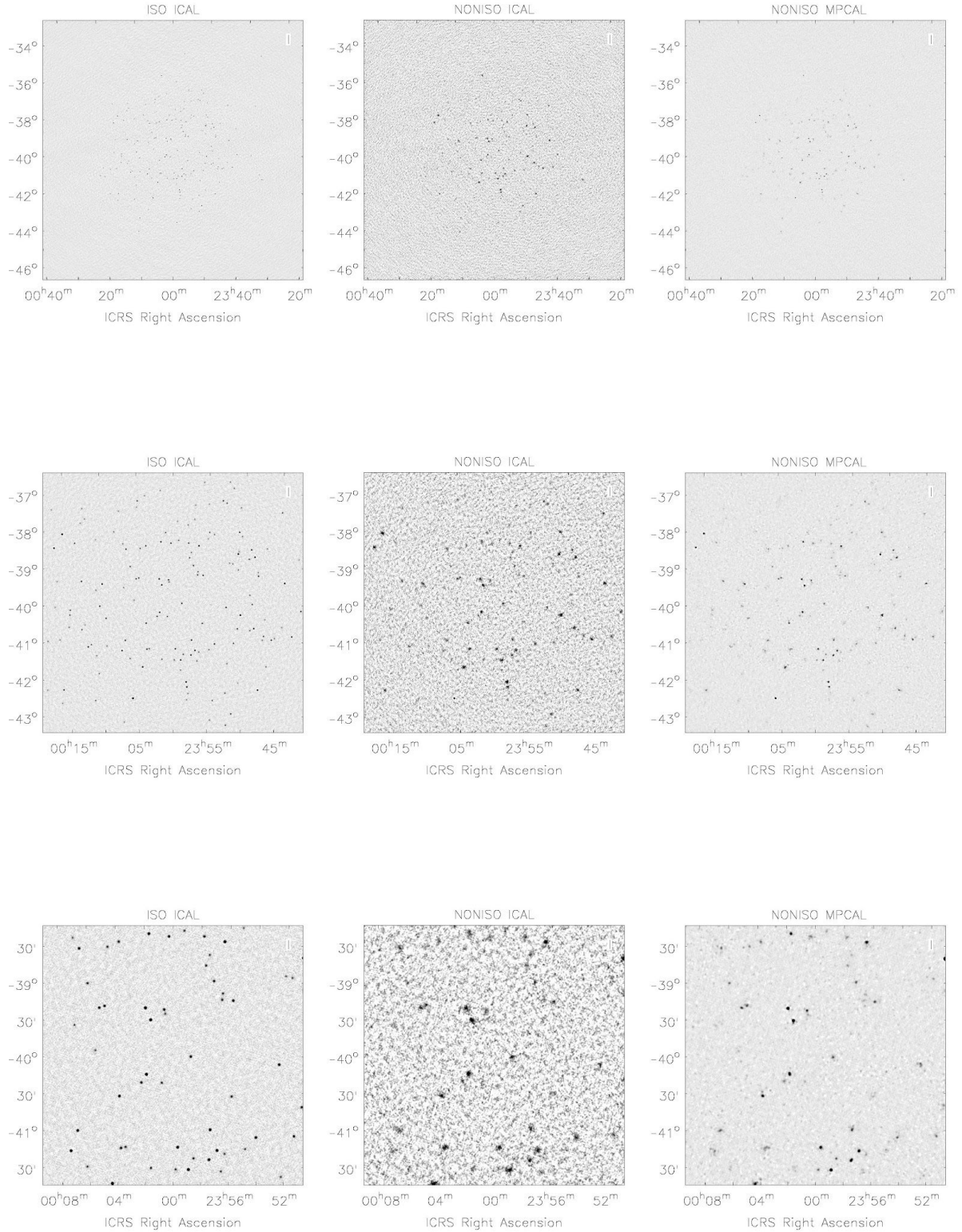
We did the following tests: running ICAL on **ISO**, running ICAL on **NONISO**, and running MPCCAL on **NONISO**. The MPCCAL facet centres were derived from the ICAL/**NONISO**, taking all sources  $> 0.5$  Jy and eliminating the weakest of pairs closer than 0.2 rad.

The resulting images are shown in Figure 7.

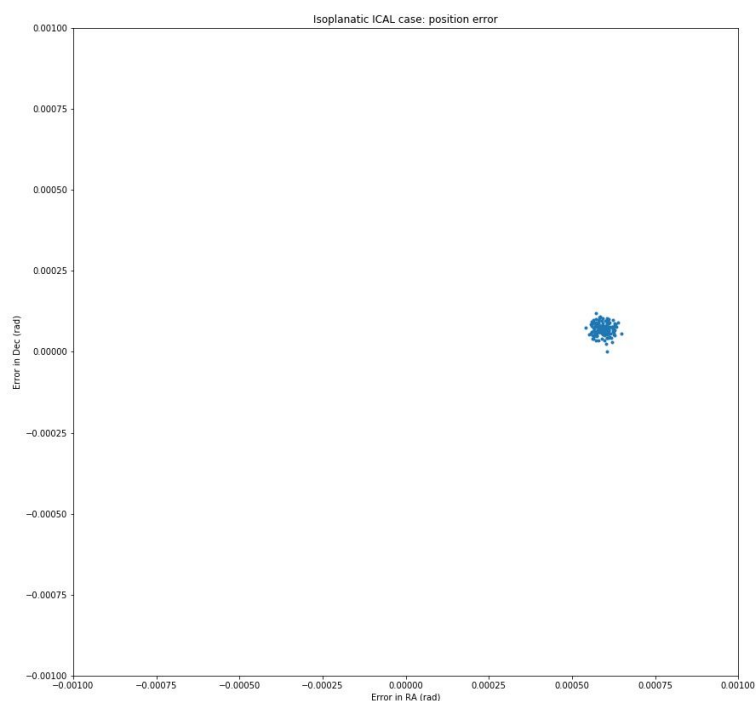
For an objective method of judging the recovery of sources, we compared the components used in the simulation with those recovered.

- Running ICAL on **ISO** produced correct point source fluxes (see Figure 9) and identical source offsets (see Figure 8). The overall position offset reflects the phase-shift of the brightest source. If we knew the location of the brightest source this position offset would be zero.
- Running ICAL on **NONISO** led to correct flux and position for only the brightest component. For all other components, the fluxes (see Figure 11) were seriously underestimated and the positions (see Figure 10) were significant wrong.
- Running MPCCAL on **NONISO** improves the recovery of sources near to the facet centres. All sources down to 0.15Jy are recovered (see Figure 13). However, the source positions show inconsistencies (see Figure 12).

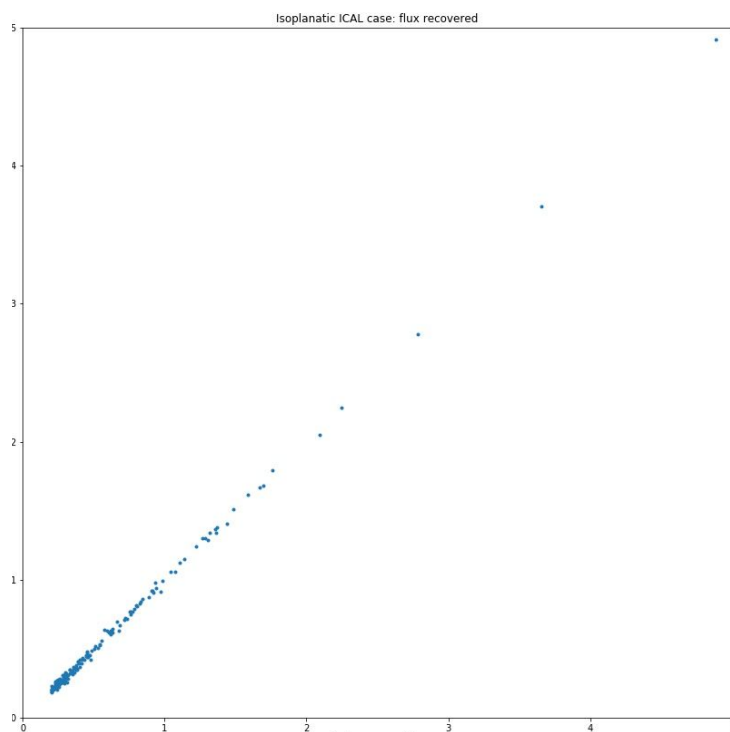




**Figure 7:** Restored images: from left to right: **ISO** with ICAL, **NONISO** with ICAL, **NONISO** with MPCAL, the zoom level increases by a factor of two on each row. The display range is -0.03, 0.3 Jy/Beam. The noise level (median deviation from absolute value) is 9.2 mJy/beam, 35 mJy/beam, 5.8 mJy/beam.

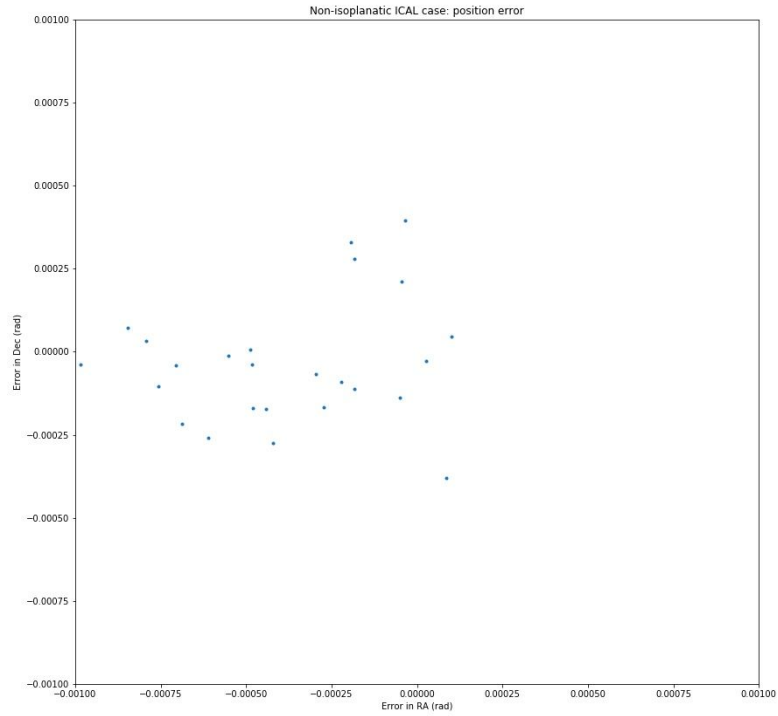


**Figure 8:** Recovered position offset as a function of simulated component flux, for isoplanatic data **ISO** processed using ICAL. For isoplanatic data, ICAL recover flux with good accuracy.

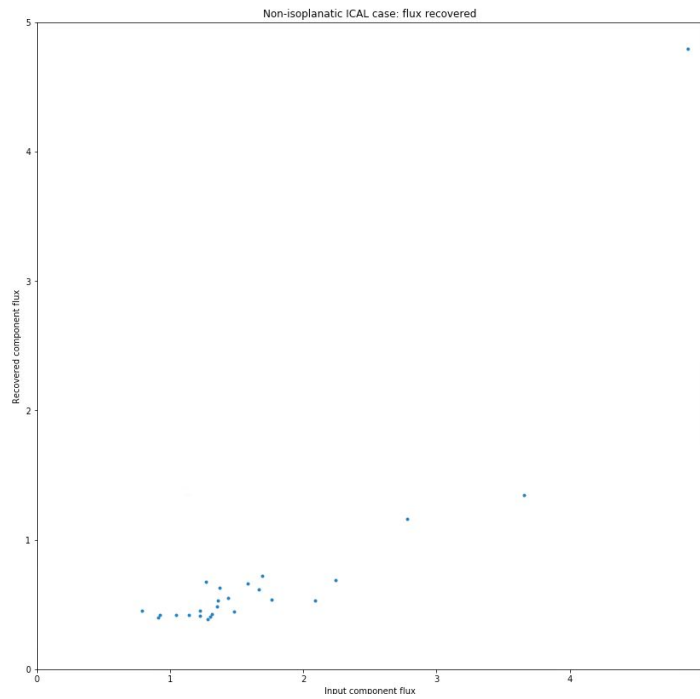


**Figure 9:** Recovered component flux as a function of simulated component flux, for isoplanatic data **ISO** processed using ICAL. For isoplanatic data, ICAL recovers flux with good accuracy.

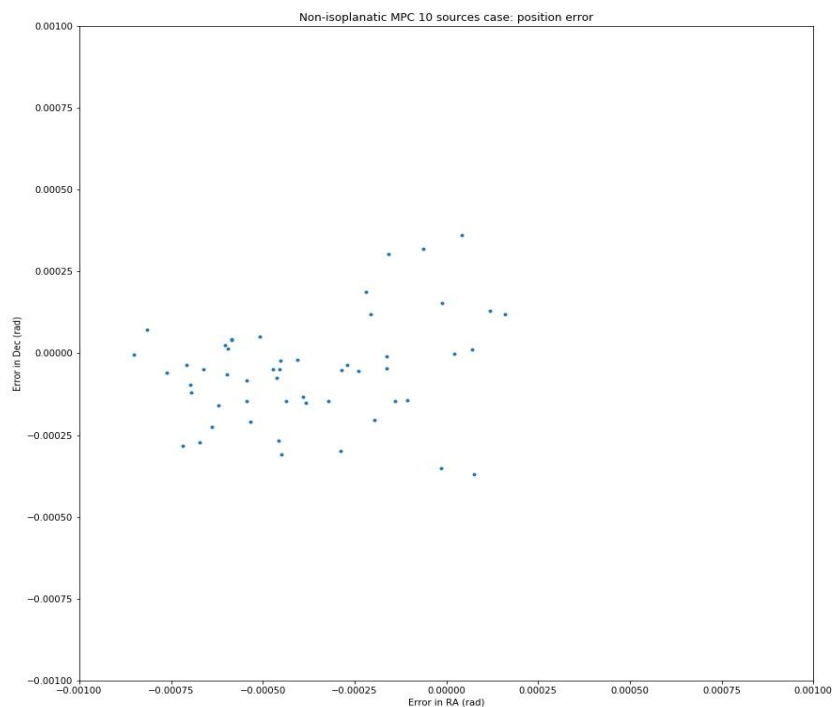




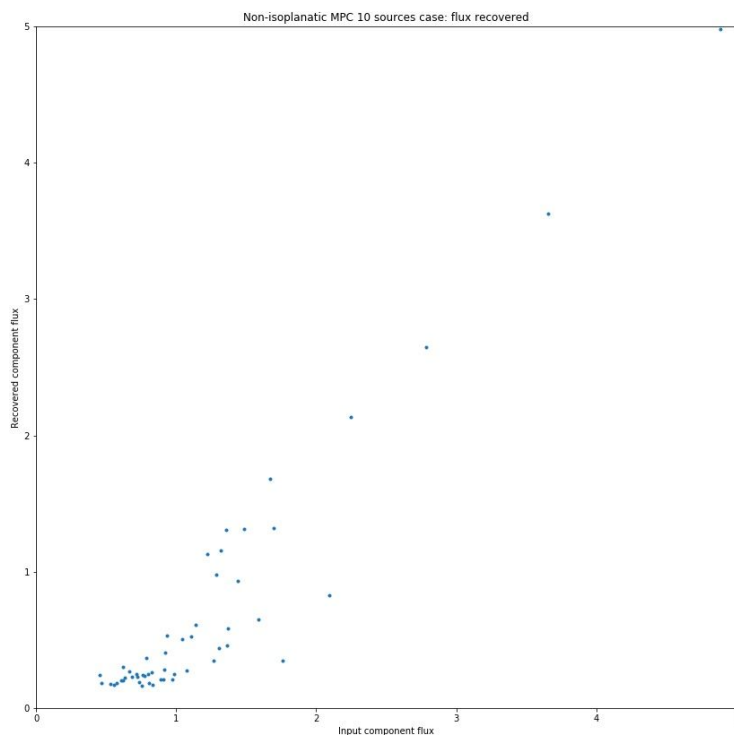
**Figure 10:** Recovered position offset as a function of simulated component flux, for non-isoplanatic data **NONISO** processed using ICAL. The positions are inconsistent as expected.



**Figure 11:** Recovered component flux as a function of simulated component flux, for non isoplanatic data **NONISO** processed using ICAL. Only the flux of the brightest source is recovered with accuracy.



**Figure 12:** Offset in recovered position as a function of simulated component flux, for non-isoplanatic data **NONISO** processed using MPCCAL. Although the noise level in the image has improved, the positions are still poorly reproduced, as expected because the phase frames of the facets are not constrained.



**Figure 13:** Recovered component flux as a function of simulated component flux, for non-isoplanatic data **NONISO** processed using MPCCAL. The bright sources (down to flux 1.5Jy) have been restored with good accuracy, but the weaker components not necessarily at the phase centre of a phase have poor recovery of the flux.

## 7 Commentary

We have demonstrated a (simple) algorithm for direction-dependent calibration using the ARL package. This algorithm is similar to Facet Calibration, and was based on the SDP Model Partition Calibration framework [RD02]. Thus we can conclude that the SDP processing architecture can in principle support direction-dependent calibration.

The additions required in ARL were:

- Skycomponent changes: filtering on flux; Voronoi partitioning; elimination of excessively close neighbours,
- Skymodel changes: addition of gaintable and mask; functions to expand by skycomponents,
- Image changes: Voronoi iterator,
- Ionospheric screen access, degriidding of the ARatmospy simulated screen, and gridding of estimated phases back to screen,
- Workflows for operating on skymodels:
  - *predict\_skymodel\_list\_arlexecute\_workflow*,
  - *invert\_skymodel\_list\_arlexecute\_workflow*,
  - *convolve\_skymodel\_list\_arlexecute\_workflow*,
- Workflow for MPC calibration: *mpccal\_skymodel\_list\_arlexecute\_workflow*, which is composed of other high level workflow including the *predict*, *invert*, and *convolve* skymodel workflows,
- Notebooks and scripts to demonstrate processing.

We have not yet demonstrated consensus optimisation. The ionospheric model suggests a possible mechanism for consensus optimisation. We can project the gain phases obtained back onto the ionospheric screen and then read back a new set of gaintable values, such as just the average phase in each phase screen cell. With the current simulation there is insufficient redundancy in the ionospheric screen sampling to make this worthwhile (see Figure 14 for an example of the sampling for ten sources).

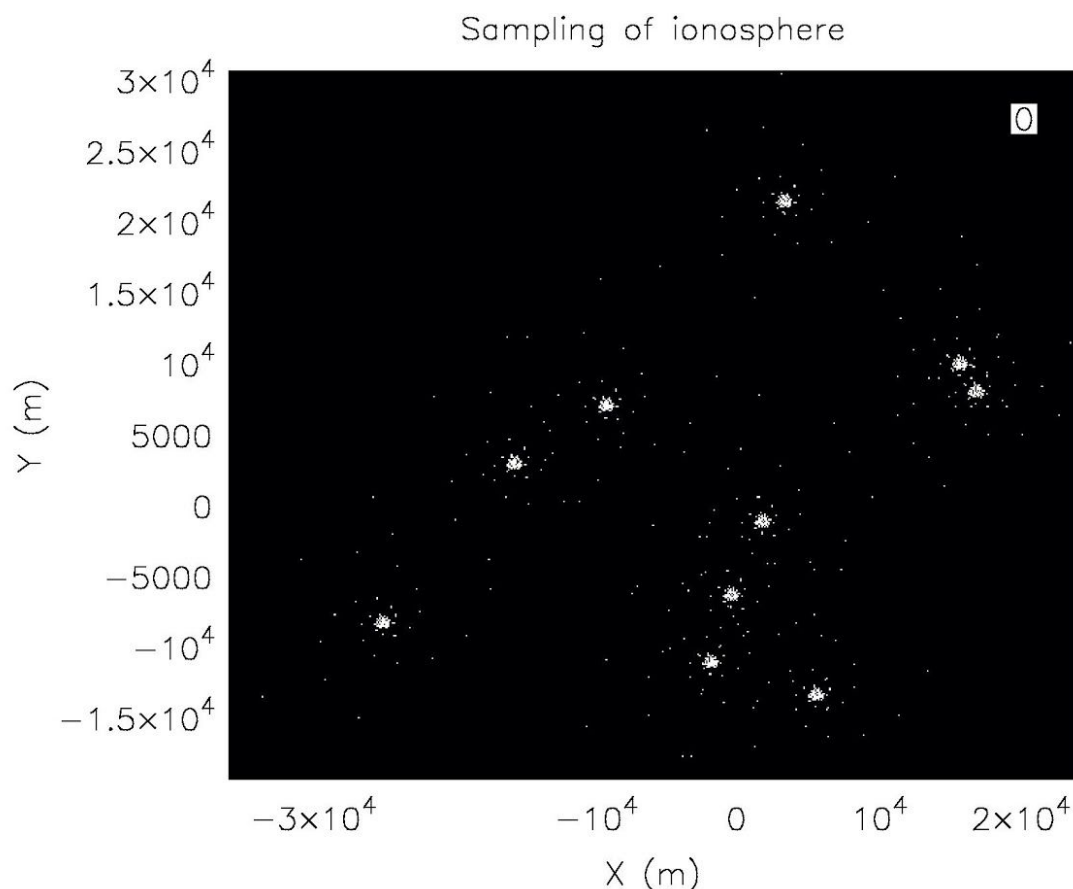
The workflows have been executed using Dask. Most of the runs have been performed for 20km maximum baseline (4K x 4K images) on an eight core 128GB desktop, though some for the entire LOW array (16K x 16K) have been done on the U.Cam CSD3 system. Scaling work has been aided by the flexibility of Dask itself, and

also by the fact that for all workflows we have serial versions, meaning that inner loops can be switched between Dask and serial versions easily.

Scaling for DD calibration and imaging is known to be challenging (see e.g. [RD04] for exposition). The both the current implementation and ARL itself have a number of optimizations possible that were not part of the scope of the current work:

- Optimal transforms of facets taking advantage of the masking (see e.g. [RD13]),
- Improved deconvolution (see e.g. [RD13], [RD14])
- Other methods of distributed cleaning (see e.g. [RD12])

These should be considered in future development.



**Figure 14:** Instantaneous sampling of the ionospheric screen at height of 300km. Each pattern reflects the array configuration.

## 8 References

- [RD01] S. Kazemi, S. Yatawatta, S. Zaroubi, P. Lampropoulos, A. G. de Bruyn, L. V. E. Koopmans, and J. Noordam, “Radio interferometric calibration

using the SAGE algorithm,” *Monthly Notices of the Royal Astronomical Society*, vol. 414, no. 2, pp. 1656–1666, Jun. 2011.

- [RD02] SDP Model Partition Calibration Workflow View
- [RD03] S. Kazemi, S. Yatawatta, S. Zaroubi, A. G. de Bruyn, L. V. E. Koopmans, and J. Noordam, *MNRAS*, vol. 414, no. 2, pp. 1656–1666, Jun. 2011.
- [RD04] S. Yatawatta, F. Diblen, H. Spreeuw, and L. V. E. Koopmans, “Data multiplexing in radio interferometric calibration,” *Monthly Notices of the Royal Astronomical Society*, vol. 475, no. 1, pp. 708–715, Mar. 2018.
- [RD05] Ionospheric Calibration of Low Frequency Radio Interferometric Observations using the Peeling Scheme. I. Method Description and First Results, H. T. Intema, et al. (2009) *Astronomy and Astrophysics* 501, 1185 ([arXiv:0904.3975](https://arxiv.org/abs/0904.3975))
- [RD06] <http://www.astron.nl/citt/facet-doc/>
- [RD07] R. J. van Weeren, W. L. Williams, M. J. Hardcastle, T. W. Shimwell, D. A. Rafferty, J. Sabater, G. Heald, S. S. Sridhar, T. J. Dijkema, G. Brunetti, M. Brüggen, F. Andrade-Santos, G. A. Ogrean, H. J. A. Röttgering, W. A. Dawson, W. R. Forman, F. de Gasperin, C. Jones, G. K. Miley, L. Rudnick, C. L. Sarazin, A. Bonafede, P. N. Best, L. Birzan, R. Cassano, K. T. Chyzy, J. H. Croston, T. Ensslin, C. Ferrari, M. Hoeft, C. Horellou, M. J. Jarvis, R. P. Kraft, M. Mevius, H. T. Intema, S. S. Murray, E. Orru, R. Pizzo, A. Simionescu, A. Stroe, S. van der Tol, and G. J. White, “LOFAR Facet Calibration,” *ApJS*, vol. 223, no. 1, p. 2, Mar. 2016.
- [RD08] C. Tasse et al, “Faceting for direction-dependent spectral deconvolution”, *A&A* 611, A87 (2018)
- [RD09] Srikar Srinath, Lisa A. Poyneer, Alexander R. Rudy; S. Mark Ammons, A computationally efficient autoregressive method for generating phase screens with frozen flow and turbulence in optical simulations, *Optics Express* 23, 33335-33349 (2015)
- [RD10] <https://github.com/shrieks/ARatmospy>
- [RD11] Hurley-Walker N., et al., GaLactic and Extragalactic All-sky Murchison Widefield Array (GLEAM) survey. I: A low-frequency extragalactic catalogue. *Mon. Not. R. Astron. Soc.*, 464, 1146-1167 (2017), 2017MNRAS.464.1146H
- [RD12] WSClean has parallel cleaning support:  
<https://sourceforge.net/p/wsclean/wiki/ParallelCleaning/>

- [RD13] Tasse, C. et al, Faceting for direction-dependent spectral deconvolution, *Astronomy & Astrophysics*, Volume 611, id.A87, 15 pp., 2018
- [RD14] An optimized algorithm for multiscale wideband deconvolution of radio astronomical images, A. R. Offringa O. Smirnov  
*Monthly Notices of the Royal Astronomical Society*, Volume 471, Issue 1, 11 October 2017, Pages 301–316