SDP Memo: The SDP Calibration Component

Document number SKA-TEL-SDP-0000029
Document type ..... REP
Revision 01C
Author ..... S. Salvini, R. Nijboer, Á. Mika
Release date 2016-04-07
Document classification ..... Unrestricted
Status ..... Draft

Lead author:

Name	Designation	Affiliation
S. Salvini	SDP.PIP.CAL Lead	University of Oxford
Signature & Date:	Stefano Salvini Stefano Salvini (Apr 6, 2016) stef.salvini@oerc.ox.ac.uk	

Released by:

Name	Designation	Affiliation
R. Bolton	SDP Project Scientist	University of Cambridge
Signature & Date:	Rosie Bolton Rosie Bolton (Apr 6, 2016) rosie@mrao.cam.ac.uk	

Version	Date of issue	Prepared by	Comments
01A	2015-05-29	S. Salvini	PDR panel comments addressed
01C	2016-04-07	S. Salvini	Issued for the ΔPDR

ORGANISATION DETAILS

Name	Science Data Processor Consortium

Table of Contents

List of Abbreviations

ADI	Alternating Direction Implicit (method)
BBS	Black-Board Self Calibration
BDA	Baseline-Dependent Averaging
CASA	Common Astronomy Software Applications
CPU	Central Processing Unit
CSP	Central Signal Processor
DD	Direction Dependent
DDE	Direction-Dependent Effects
DFT	Discrete Fourier Transform
DI	Direction Independent
EM	Expectation Maximisation
FFT	Fast Fourier Transform
FLOPS	Floating Point Operations per Second
HBA	High-Band Array
GPU	Graphics Processing Unit
I/O	Input/Output
LMA	Levenberg-Marquardt Algorithm
LMC	Local Monitor and Control
LOFAR	Low Frequency Array
LS	Least Squares
LSM	Local Sky Model
LTS	Local Telescope State
ML	Maximum Likelihood
MPI	Message Passing Interface
NDFT	Non-uniform Direct Fourier Transform
PDR	Preliminary Design Review
SAGE	Space Alternating Generalised Expectation Maximisation
SDP	Science Data Processor
SKA	Square Kilometre Array
SKAO	SKA Office
StEFCal	Statistically Efficient and Fast Calibration
TBD	To Be Defined
TM	Telescope Manager

List of Symbols

$..^\dagger$	A superscript dagger (†) denotes Hermitian conjugation:for example $M^\dagger$ is the Hermitian conjugate of the matrix M
$A_{a,s}$	2×2 matrix of atmospheric/ionospheric effects. Depends on both source and receiver (line of sight through atmosphere/ionosphere).
a, b	Receiver indices
B_s	Source brightness 2×2 matrix (with $N_{\rm pol}$=2)
C_s	Cholesky factor (2×2 matrix).
c	Speed of light.
$D_{a,s}$	Projection of the source onto receiver dipoles (2×2 non-unitary matrix). Depends on both receiver and source. (Receiver beam pattern).
f	Observation frequency.
G_a	Receiver complex gain 2×2 matrix. Depends only on the receiver.
$\mathcal{G}$	Block-diagonal matrix with the gain 2×2 matrices G_a along the diagonal.
$K_{a,s}$	Geometric phase. Constant diagonal 2×2 matrix (i.e. diagonal elements are identical). Depends on both receiver and source.
$N_{\rm beam}$	Number of beams simultaneously observed by the array
$N_{\rm ch}$	Number of channels
$N_{\rm dir}$	Number of directions
$N_{\rm it}$	Number of iterations required for iterative methods (a notional place holder)
$N_{\rm major}$	Number of major CLEAN cycles (within each SelfCal loop if appropriate)
$N_{\rm pix}$	Number of pixels along one axis of a grid
$N_{\rm pol}$	Number of polarisations (assumed to be 2)
$N_{\rm rec}$	Number of receivers (2 dipoles for each receiver/station)
$N_{\rm SelfCal}$	Number of SelfCal loops
$N_{\rm source}$	Number of sources
$N_{\rm vis}$	Size of the visibility matrix (total number of receivers): $N_{\rm vis} = 2N_{\rm rec}$ for full polarisation (as in this document)
P_s	Sky corruption (2×2 matrix), independent of receiver
$Q_{a,s}$	Product of Jones matrices ($Q_{a,s} = K_{a,s}D_{a,s}A_{a,s}$)
s	Source index
$t_{\rm dump}$	Correlator dump time (in seconds)
$t_{\rm obs}$	Total observation time (in seconds)
V	Model Visibility matrix (observed)
V^M	Model sky visibility matrix
V^C	Corrected visibility matrix
λ	Observation wavelength: $\lambda = \frac{c}{f}$.

List of Tables

Summary

This document describes the components required by the calibration within the Science Data Processor (SDP) processing pipelines. It also provides estimates of the computational costs in terms of flops (total number of operations required) and FLOPS or flops/second, i.e. the computational power required to be actually delivered in order to make the processing possible. The data and metadata required for the calibration are also listed.

It should be noted that, to date, a full calibration strategy has not been delivered by the SKA Office (SKAO). This has led, necessarily, to some voids within and limitations of this document.

Also, a number of topics are very much likely to be of importance for the SKA1, and increasingly so during the life of the instrument, although neither theory nor practice are adequately established to provide a full treatment. None the less this document addresses them from a qualitative point of view at least to acknowledge their relevance in the life cycle and continuous development of the SKA1.

Applicable and Reference Documents

Applicable Documents

The following documents are applicable to the extent stated herein. In the event of conflict between the contents of the applicable documents and this document, *the applicable documents* shall take precedence.

Reference Number	Reference
AD01	SKA-TEL-SDP-0000013 – SDP Element Architecture Design
AD02	SKA-TEL-SDP-0000027 – SDP Pipelines Design
AD03	SKA-TEL-SDP-0000040 – SDP Parametric Modelling document

Reference Documents

The following documents are referenced in this document. In the event of conflict between the contents of the referenced documents and this document, *this document* shall take precedence.

Reference Number	Reference
RD01	Boyd S., Parikh N., Chu E., Peleato B., Eckstein J. (2011): Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Foundations and Trends in Machine Learning: 3, 1-122.
RD02	Fessler, J. and Hero, J. A. (1994): Space-alternating generalised expectation-maximisation algorithm. IEEE Transactions on Signal Processing , 42, 1656.
RD03	Kazemi, S., et al. (2011): Radio interferometric calibration using the SAGE algorithm. MNRAS (414), 1656.
RD04	Nocedal, J. and Wright, S. (2006): Numerical Optimization. Springer.
RD05	Salvini, S. and Wijnholds, S. (2014): Fast gain calibration in radio astronomy using alternating direction implicit methods: Analysis and applications. A&A , 571.
RD06	Salvini, S. and Wijnholds, S. (2014): StEFCal An Alternating Direction Implicit method for fast full polarization array calibration. General Assembly and Scientific Symposium (URSI GASS). Beijing.
RD07	Yatawatta, S. et al. (2009): Radio interferometric calibration using the SAGE algorithm. IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, pp. 150, Piscataway, NJ, USA.
RD08	van Weeren, R.J. et al. (2016): Lofar Facet Calibration. arXiv:1601.05422
RD09	Yatawatta, S. (2015): Distributed radio interferometric calibration. MNRAS (449), 4506-4514
RD10	Smirnov, O.M. (2011): Revisiting the radio interferometer measurement equation. MNRAS (414), 1656-1666.
RD11	Smirnov, O.M. and Tasse, C. (2015): Radio interferometric gain calibration as a complex optimization problem. MNRAS (449), 2268-2684.

Reference Number	Reference
RD12	Noordam, J.E. (2004): LOFAR calibration challenges. Proc. SPIE 5489, Ground-based Telescopes, 817; doi:10.1117/12.544262.
RD13	Cornwell, T.R. (2008): Multiscale CLEAN Deconvolution of Radio Synthesis Images. IEEE Journal of Selected Topics in Signal Processing (2), 793-801.
RD14	Yatawatta, S. (2011): Shapelets and Related Techniques in Radio-Astronomical Imaging. URSI (http://www.ursi.org/proceedings/procga11/ursi/J06-6.pdf).
RD15	Golub, G.H. and Van Loan, C.F., (1996): Matrix Computations (3rd Edition). The Johns Hopkins University Press, Baltimore, ISBN 0-8018-5414-8.

1 Introduction

This document describes the Calibration Component of the SDP data processing pipelines. It supports the SDP PDR documentation set and in particular the Pipelines Design document AD02 and the Parametric Modelling document AD03.

Figure 1 from the Pipeline Design document (AD02) shows the position within the functional/data flow diagram of the calibration described in this document, in the “Calibrate and Image” and “Calibrate Real Time” components.

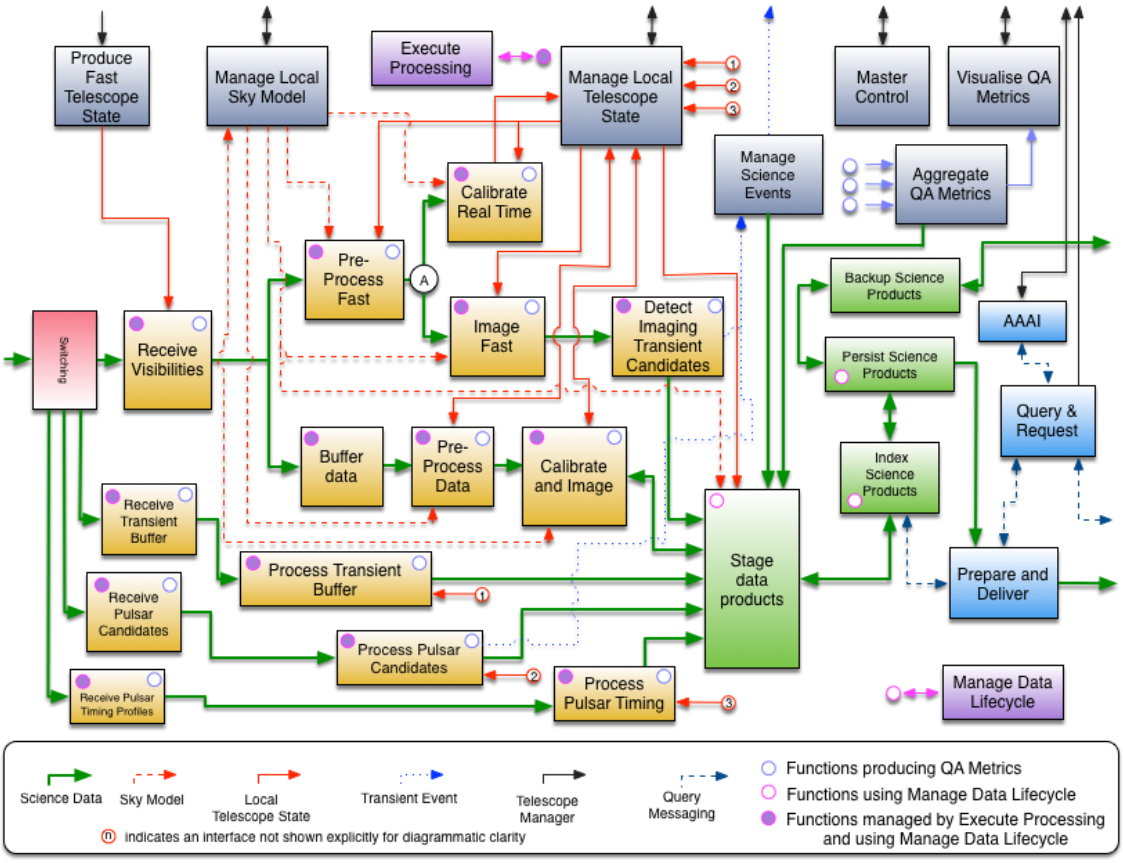

Figure 1: SDP functional and data flow at L2, from AD01, showing the SDP functions at L2 and the flow of data and information between them. Arrows indicate the interfaces over which data, metadata or control information passes and the directions of the flow: green arrows indicate data flow, red arrows control and/or metadata, blue event information, black interfaces to the Telescope Manager (TM) and dark blue Deliver Data messaging. Solid and dashed lines have no additional meaning and are used only to help in the visual presentation. The control and metadata interfaces for Manage Data Lifecycle and Aggregate QA Metrics are shown using colour-coded circles within function blocks. The block colours provide a cross reference to the functional decomposition at L1 (see AD01) including the subdivision of Process Data into Process Data: Platform, Pipelines and Execute Processing. The Execute Processing Function has an interface to all of the Process Data: Pipelines functions. The numbers within red circles are used to show the linkage between some functions and the Manage Local Telescope State Function. The scope of this document are the yellow blocks “Calibrate and Image” and “Calibrate Real Time” in this diagram.

Figure 2, also from AD02, shows the relative position and connection between the components of the calibration described in section 2 in the case of real-time calibration.

It should be stressed that the overall calibration strategy should have been provided by the SKAO. However, at the time of writing, this has not yet happened. Calibration and Processing Strategies are as yet unknown and we could expect them to take some time to develop, possibly into the Commissioning and Operational phases, thus requiring a general and flexible approach.

Therefore, in order to make progress, we had to make some assumptions, while at the same time we are conscious of the necessary limitations of this document. These are discussed below in subsection 1.1.

Currently, it is envisaged that calibration would be carried out within the SDP computing facilities. However, some parts of the calibration process could be co-located with or indeed moved to the Central Signal Processor (CSP), in order to reduce data ingest rates, provided other items, such as cost, maintenance, etc. did not worsen.

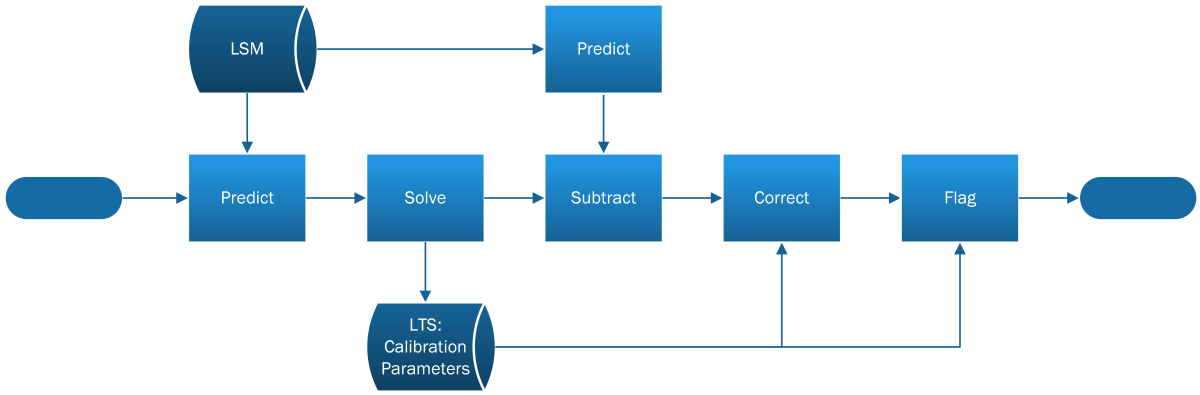

Figure 2: A functional and data flow breakdown of the F1.4 Calibrate Real-time function. Calibration solutions are produced for use in other parts of the SKA system, e.g. the CSP Beamformer or the SDP F1.5 Image Fast function. Optionally, foreground sources are subtracted from the visibilities, visibilities may be corrected for Direction Independent effects, and visibilities may be flagged based on outlier calibration solutions.

This document is organised as follows. Subsection 1.1 in this Introduction discusses the assumptions and constraints; this is followed by a clear definition of the computational costs and how these are computed (section 1.2).

The next section (no. 2) then discusses the operations required and associated costs for the canonical Direction Independent (DI) calibration.

Section 3 discusses aspects of the Direction Dependent (DD) calibration, global solvers and extended sources, respectively, which are likely to play an important role in data processing but for which neither theory nor practice are sufficiently advanced to allow but a discussion in qualitative terms.

After a section listing the likely data and metadata required by the calibration stages (no. 6), the final section (no. 7) applies the operations count to the practical examples of the SKA1-LOW and MID instruments.

1.1 Considerations, Assumptions and Limitations of this Document

It should be stressed that because of the nature of the data reduction, processing will be most likely memory-bound (and I/O bound) rather than compute-bound. Leaving aside I/O (under the surmise that data will be loaded as few times as possible into the computational engine), the density of operations that can be applied for items of data loaded from memory is likely to be low. Thus computational times will be dominated by data (memory) access rather than by flop counts. Indeed, the ratio between computational capabilities of processors and bandwidth from memory has been widening in the last few years.

The amount of data to be transferred to/from memory does not represent the whole story. Algorithms and their pattern of access to memory, as well as details of the hardware have important roles to play.

Total data size is not a good indicator of performance either. For example, a matrix-vector product such that the matrix cannot fit in cache requires a complete data refresh at each iteration: so any iterative methods relying on that are affected (such as Krylov methods, Conjugate Gradient, etc.).

These effects are extremely difficult to analyse, predict and model and are very much hardware dependent.

For example, 2D FFTs are going to have a major impact in the SKA computational economy. Because of their importance in many fields, hardware vendors strive to produce highly optimised versions on their platforms. However, their efficiencies (i.e. FLOPS returned against peak performance) are between 5% to 15% for the problem sizes of interest to the SKA.

Different components of the pipelines could have different complexities and incur different computational costs. It would seem attractive to be able to rely on different platforms or nodes tailored to different tasks. However, benefits could be reduced by needing to move or duplicate data on different nodes (data movement being identified as a major constraint in the SDP platform). Furthermore, separate nodes would need to be procured and maintained.

It should be noted, however, that currently the computational engines being considered are mostly hybrid, consisting of different components, such as CPU/GPU nodes (or similar). Calibration could be carried out on a component, say the CPU, while the GPU carries out concurrently some other tasks (as it has been suggested for the calibration of SKA1-LOW stations).

Since the additional computational complexity, critical paths, difficult load balancing etc. would make the use of different nodes for each pipeline task rather difficult, it is most likely that different tasks would be performed on different components of a hybrid engine/node.

Computational costs are reported in number of operations required and have been obtained under a number of assumptions:

Calibrations occur at every cross-correlator dump. All calibrations are carried out at the same pace as cross-correlation data. This corresponds to a worst case scenario which we need to adopt until further information is made available.

We only consider full polarisation. Brightness matrices and Jones matrices are all defined in terms of 2×2 blocks. It should be noted that should only the unpolarised signal be

considered, for the same number of dipoles per receiver, the operation count would be halved.

We use the Jones matrix formalism. We used it throughout this document: it is entirely equivalent, at least in line of principle, to using a Mueller based approach (although the operation count may differ according to the specific implementation). The relationship between Jones and Mueller formalisms is very briefly described in appendix A.

We assume double precision throughout. This may not be strictly needed, however it is consistent with the worst case scenario, and it effects only bandwidth requirements (not flops).

No flagging is considered. It is assumed that all cross-correlation dumps and all visibilities are correct (no flagging required).

Auto-correlation or no auto-correlation? With autocorrelations, many operations would scale with $\frac{N_{\rm rec}(N_{\rm rec}+1)}{2}$, without it as $\frac{N_{\rm rec}(N_{\rm rec}-1)}{2}$. In both cases the difference with using simply $N_{\rm rec}^2/2$ is $N_{\rm rec}^{-1}$. This is about $1/197$ for MID and $1/512$ for LOW: given the uncertainty in the number of operations this can certainly be ignored. Hence, the question of whether autocorrelation terms are or are not used is of no relevance within the scope of this document.

Some aspects that are outside the scope of this document are:

Bandpass calibration: The bandpass will have a time and frequency structure at a wide range of scales. The complexity incurred could thus be potentially high. Bandpass calibration is beyond the scope of this document.

Averaging, BDA, etc.: The effects of averaging and baseline-dependent averaging (BDA) cannot be quantified at this time. However, the effects of some compression due to averaging are included in the examples.

Direction Dependent Effects (DDE): Effects due to the ionosphere, pointing errors, etc. have been ignored at this stage. They are still being investigated by LOFAR. DDE calibration is not sufficiently established either in practice or theory for a quantitative analysis, but it will be qualitatively discussed in section 3.

We also report in this document on the data and metadata required by the calibration process, which we have heuristically classified according to the following criteria:

Data: Items of information following the main flow from the cross-correlator to the SDP Engine at the dump rate. Hence, the injection and retrieval of these items are synchronous with the computation.

Metadata: Information qualifying the computations to be carried out, including computation type, parameters, etc. This information can be acquired asynchronously.

1.2 On flops and FLOPS (Computational Costs)

Originally, *FLOPS* was the acronym of **FL***oating-point* **OP***erations per* **S***econd*. However, in current use, a *flop* is used by the Numerical Analysts and Computational Scientists communities as the number of equivalent real operations, *irrespective of time* (see for example RD15). Hence, throughout this paper and with no exception, the following conventions are used

flops Real floating point operations.

FLOPS = flops/second Number of flops that are, can or should be computed per second.

Currently, in the SKA SDP computational costs are measured by the number of real flops. In accordance to common practice among numerical analysts and computational scientists, a flop is an individual operation; note that a real mult-add consists of 2 FLOPS. In terms of real operations, complex floating point operations have the following costs (see Table 1):

Operation	Nr. of flops
$a + b$	2
$a - b$	2
$a \cdot b$	6
$a \cdot b + c$ (mult-add)	8

Table 1: Operation costs of complex, floating point operations.

It is also customary for numerical analysts to attribute the cost of 1 flop for storing numbers, e.g. when not using them after an operation, so that a write-back to memory is implied. The debate about using 6 rather than 8 for complex multiplications rages on: it is very unlikely that complex multiplications would be used in isolation without a sum (for a full fused mult-add) or without a write-back to memory (the author believes that it should be 8). However, it makes very little difference in the overall costs.

We will also report only the leading term of the number of operations. Lower order terms appear to gain in importance for smaller problem sizes. However, for smaller problems a number of other effects play such a role that the flops count is not indicative of potential performance. For example:

- Typically, a mult-add requires between 15 to 20 stages to be carried out. Pipelining (i.e. overlapping the stages for subsequent operands) allows the delivery of one result per cycle after the pipe has been filled. For example, if 1,000 mult-adds have to be carried out, and the number of stages is 20, the number of cycles required would be 1,020. The number of flops delivered (at 8 flops per mult-add) per cycle would be rather close to 8. However if only 10 mult-adds were required, the same flops per cycle would amount to $\frac{8 \cdot 10}{10+20} = 2.7$, and the number of operations would be a bad predictor of performance. This is, if anything, further emphasised by modern vectorised cores.
- Also, cache and memory residence would cause very non-linear effects, which render the analysis of small problems very different from larger problems.

For example, the solution of a Positive Definite linear system of equations would require Cholesky factorisation, followed by the solution of two triangular systems of equations.

Operation	Nr. of flops
Cholesky Factorisation	$\frac{1}{3}N^3 + \mathcal{O}(N^2)$
Solution of triangular systems	$2N^2 + \mathcal{O}(N)$

Table 2: Example of operations count: solution of a Positive Definite System of Equations

The operation costs are the following (see Table 2):

Unless the sum of the coefficients of the $\mathcal{O}(N^2)$ terms is of the same order of magnitude as $\frac{1}{3}N$, adding lower order terms would only add a small contribution to the overall estimate, hence the solution is reported to require $\frac{1}{3}N^3$ flops. But this is, naturally, only part of the story: some parts of the computation could require fewer flops but have more memory access: this would require a deeper analysis than simple operation count would support.

Reporting only the higher order flop count is also common practice for numerical analysts and computational scientists, exceptions do of course exist for a number of reasons, as mentioned.

2 Direction Independent Calibration: Computational Costs

In the Measurement Equation formalism, we define the model sky visibility element ${V^M}_{a,b}$ (itself a 2×2 matrix) as:

$${V^M}_{a,b} = \sum_{s=1}^{N_{\text{source}}} K_{a,s} D_{a,s} A_{a,s} P_s B_s {P_s}^\dagger {A_{b,s}}^\dagger {D_{b,s}}^\dagger {K_{b,s}}^\dagger \tag{1}$$

where the summation extends over the sources in the model sky and a and b are receiver indices. Note that antenna gains (instrument time-dependent corruptions) are not included (see below). Basically the model sky visibility corresponds to the signal received by the interferometer when all receivers have identical, in-phase gains.

In eq. (1) the source brightness (coherence matrix) is (when using a linear polarisation basis):

$$B_s = \begin{pmatrix} I_s + Q_s & U_s + \mathrm{i}V_s \\ U_s - \mathrm{i}V_s & I_s - Q_s \end{pmatrix} \tag{2}$$

and the other Jones matrices in the equations, equally 2×2 matrices are: P_s, the sky corruption independent of the receiver; $A_{a,s}$, the atmospheric/ionospheric effects (depends on both the source and the receiver, i.e. the line of sight through the atmosphere/ionosphere); $D_{a,s}$, the projection of the source onto the receiver dipoles (2×2 non-unitary matrix), which depends on both the receiver and the source (i.e. the receiver beam pattern); $K_{a,s}$, the geometric phase, a constant diagonal 2×2 matrix (i.e. the diagonal elements are identical) which depends on both receiver and source.

The aim of the calibration process is to obtain estimates for the complex gains G_a (2×2 matrices) which depend only on the receiver and minimise in some norm the difference between the observed and the model sky:

$$\min_{\mathcal{G}} \| V - \mathcal{G}\, V^M\, \mathcal{G}^\dagger \|_F \tag{3}$$

where $\mathcal{G}$ is a block-diagonal matrix with G_a along the diagonal:

$$\mathcal{G} = \mathrm{diag}(G_a) = \begin{pmatrix} G_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & G_{N_{\mathrm{rec}}} \end{pmatrix}$$

Given the gain solutions, several scenarios for correcting the visibilities for these gains exist. For example, some sources can be subtracted from the gains before correcting. This is done by corrupting the model sky visibilities by the gains and subtracting them from the visibilities:

$$V^D = V - \mathcal{G}\, V^M\, \mathcal{G}^\dagger \tag{4}$$

and finally the observed visibilities can be corrected by the inverse gains:

$$V^C = \mathcal{G}^{-1} V^D\, \mathcal{G}^{-\dagger}. \tag{5}$$

Essentially, Direction Independent calibration (see also subsection 1.1) consists of four stages:

Predict (Model Sky Visibilities): This may involve applying Jones matrices to the model sky sources from previously obtained information. The model sky visibilities can be finally computed by 3-D NDFT (Non-uniform Direct Fourier Transform) as in eq. (1).

Solve: Receivers' complex gains are estimated by χ^2 minimisation between the observed and model visibilities, as in eq. (3).

Subtract (of the corrupted model sky visibilities): The model sky visibilities are corrupted by the computed gains, then subtracted from the observed visibilities, as in eq. (4).

Correct: The observed visibilities, after subtraction, are then corrected by applying the inverses of the gains as in eq. (5).

2.1 Computational Costs of Applying Jones Matrices

Each Jones matrix is 2×2 complex. Some Jones matrices are diagonal. It is useful to classify the Jones matrices in terms of their nature and their operations. This can best be done in matrix form. For each source we define a block diagonal matrix whose diagonal is given by the 2×2 Jones matrices for all receivers. For example, given generic Jones matrices $U_{a,s}$ (which depend on both receiver and source) and V_s (which depend only on the source) we define:

$$\mathcal{U}_s = \mathrm{diag}(U_{a,s}) = \begin{pmatrix} U_{1,s} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & U_{N_{\mathrm{rec}},s} \end{pmatrix}$$

$$\mathcal{V}_s = V_s \otimes I_{N_{\mathrm{rec}}} = \begin{pmatrix} V_s & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & V_s \end{pmatrix}$$

and if $J_{N_{\mathrm{rec}}}$ is the matrix of ones (all one entries) of order N_{rec} nominally, we can define

$$\mathcal{B}_s = B_s \otimes J_{N_{\mathrm{rec}}} = \begin{pmatrix} B_s & \cdots & B_s \\ \vdots & \ddots & \vdots \\ B_s & \cdots & B_s \end{pmatrix}$$

from which we can obtain the full visibility matrix for each source s in terms of these matrices:

$$V^M = \sum_{s=1}^{N_{\rm source}} \mathcal{K}_s \mathcal{D}_s \mathcal{A}_s \mathcal{P}_s \mathcal{B}_s \mathcal{P}_s^\dagger \mathcal{A}_s^\dagger \mathcal{D}_s^\dagger \mathcal{K}_s^\dagger \tag{6}$$

It is now straightforward from eq. (6) and exploiting the block diagonal nature of the components in it to produce a table listing the number of operations required. Table 3 reports the number of flops required to apply Jones matrices of various types to another Jones matrix (assumed to be complex) and to a general complex matrix, such as the visibility matrix. This includes the summation over all the sources. It should also be stressed that applying Jones matrices from the right or from the left requires the same number of operations.

Jones Matrix type	**Apply to Jones Matrix Number of flops**		**Apply to General Matrix Number of flops**	
	Complex Jones	Real Jones	Complex Jones	Real Jones
Full	$64\,N_{\rm rec}\,N_{\rm source}$	$32\,N_{\rm rec}\,N_{\rm source}$	$64\,N_{\rm rec}^2\,N_{\rm source}$	$32\,N_{\rm rec}^2\,N_{\rm source}$
Diagonal	$32\,N_{\rm rec}\,N_{\rm source}$	$16\,N_{\rm rec}\,N_{\rm source}$	$32\,N_{\rm rec}^2\,N_{\rm source}$	$16\,N_{\rm rec}^2\,N_{\rm source}$
Constant diagonal	$32\,N_{\rm rec}\,N_{\rm source}$	$16\,N_{\rm rec}\,N_{\rm source}$	$32\,N_{\rm rec}^2\,N_{\rm source}$	$16\,N_{\rm rec}^2\,N_{\rm source}$

Table 3: Operation count for the product of a complex (columns 2 and 4) or real (columns 3 and 5) Jones matrix with a complex Jones matrix (columns 2 and 3) or general complex matrix (columns 4 and 5).

2.2 Predict (generate model sky visibilities)

The model visibilities can be computed, for all the sources in the model sky, by using eq. (1). In general, it is not possible to give an estimate of the computational costs for the various Jones matrices appearing in the equation and we shall not try to do so here, with the exception of the geometric phase factors $K_{a,b}$, which represent the location of the receivers, hence can be used to compute the baselines phases for each source.

Under the assumptions that the dipoles for the a-th receiver are co-located at x_a, y_a, z_a, and that:

$$K_{a,s} = e^{i\frac{2\pi}{\lambda}(l_s x_a + m_s y_a + n_s z_a)} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{7}$$

where l_s, m_s, n_s are the s-th source direction cosines, the computation can be carried out, to minimise costs, in the sequence:

- Scale either the receiver coordinates or the source direction cosines (whichever is smaller), by $\frac{2\pi}{\lambda}$. We will assume that $N_{\rm rec} < N_{\rm source}$.
- Compute all $q_{a,s} = l_s \tilde{x}_a + m_s \tilde{y}_a + n_s \tilde{z}_a$ arguments of the complex exponentials.
- Compute $\sin q_{a,s}$ and $\cos q_{a,s}$.

Note that sines and cosines are computed and not complex exponentials. The reason is that on most platforms, there are versions of sines and cosines that are pipelineable, requiring

Operation	Nr. of flops
Scale the receiver coordinates	$6\,N_{\rm rec}$
Compute the trigonometric functions	$10\,N_{\rm rec}\,N_{\rm source}$

Table 4: Predict (generate model sky visibilities): operations count.

a clock cycle each; complex exponentials, in general, cannot be pipelined. The compute costs are shown in Table 4.

Lower order terms in the number of operations have been neglected. As the data can be accessed sequentially in memory, we would expect good efficiency.

We are therefore in a position to compute the overall operation count for the predict step.

First we compute for each receiver and each source (the order of the operations is of course important for efficiency but irrelevant for the operations count):

$$Q_{a,s} = K_{a,s} D_{a,s} A_{a,s} P_s \tag{8}$$

Based on Table 3 the operation count is linear with $N_{\rm rec}$ and is given by $\sim 192\,N_{\rm rec}$, certainly lower than other terms, thus it can be ignored.

Rewriting eq. (1) in terms of $Q_{a,s}$:

$$V^M{}_{a,b} = \sum_{s=1}^{N_{\rm source}} Q_{a,s}\, B_s\, {Q_{b,s}}^\dagger \tag{9}$$

which shows that we need to apply a Jones matrix to the coherence matrix from the left and from the right.

Alternatively, Jones matrices could be applied only one-sided, and the final assembly of the visibility matrix carried out at the end. In fact, for each source, B_s is almost certainly Hermitian positive definite. Hence we can carry out a Cholesky factorisation:

$$B_s = C_s {C_s}^\dagger$$

Exceptions can occur, for example if a source is completely polarised in a specific direction. In such cases B_s can be positive semidefinite. Such exceptions, likely to be few in number, can be treated separately, adding little to the overall computational load $\mathcal{O}(\underline{N_{\rm source}} N_{\rm rec}^2)$ operations where $\underline{N_{\rm source}} \ll N_{\rm source}$ denotes the number of "exceptional" sources in this sense.

Assuming that all B_s are positive-definite, we can write:

$$V^M{}_{a,b} = \sum_{s=1}^{N_{\rm source}} Z_{a,s}\, {Z_{b,s}}^\dagger \tag{10}$$

where:

$$Z_{a,s} = Q_{a,s}\, C_s$$

The resulting number of operations is shown in Table 5, where only one half of the visibility matrix is computed because of its Hermitian properties.

Operation	Total nr. of flops
Compute $K_{a,s}$	$10\,N_{\rm rec}\,N_{\rm source}$
Compute C_s	$\frac{32}{3}N_{\rm source}$
Compute $Q_{a,s}$	$(64+64+64+32)\,N_{\rm rec}\,N_{\rm source} = 224\,N_{\rm rec}\,N_{\rm source}$
Compute $Z_{a,s} = Q_{a,s}\,C_s$	$32\,N_{\rm rec}\,N_{\rm source}$
Compute the visibilities two-sided	$\approx 64\,N_{\rm rec}^2\,N_{\rm source} + \mathcal{O}(N_{\rm rec}\,N_{\rm source})$
Compute the visibilities one-sided	$\approx 32\,N_{\rm rec}^2\,N_{\rm source} + \mathcal{O}(N_{\rm rec}\,N_{\rm source})$

Table 5: Predict operation count

Table 5 shows that the actual assembling of the visibility matrix is the expensive step, other costs being negligible. The one-sided approach does reduce the number of operations by a factor two, at the cost of potential loss of stability, caused by the condition number of the brightness matrices B_s: this could be severe in the case of very highly polarised sources.

If the number of sources to be predicted is very large, the model sky visibilities could be obtained from the FFT on an image, followed by degridding. The cost of this procedure is independent of the number of sources, but depends on the number of pixels ($N_{\rm pix}^2$) of the model image as $5N_{\rm pix}^2\log_2(N_{\rm pix})$ + degridding cost: this could be prohibitively expensive at the cadence of the cross-correlator dumps. It should be noted that in the case of LOFAR, source models of 10,000 components may be encountered and this approach may prove useful. The FFT + degrid approach can also be used to resolve issues arising from extended sources (see section 5).

2.3 Solve (minimise the difference between the model and the observed visibilities)

This step aims to estimate the complex gains through the minimisation in eq. (3), a non-linear minimisation process. Although the L_2 or Frobenius norms have been traditionally used, other minimisation norms could be used in the future, such as the L_1 norm. This corresponds to a different error distribution.

We discuss here very briefly the ubiquitous Levenberg-Marquardt method against the more recent StEFCal. The latter has now shown that the costs of calibration do not need to dominate the computational budget for the reduction of radio astronomical observations and could represent only a minor contribution.

It should be highlighted that all algorithms used are local solvers, in the sense that they cannot identify a global minimum for the function being optimised: unless further assumptions are made, in general the minimisation problems in the case of full polarisation are not necessarily convex. These methods can be used independently for individual or groups of channels, time slots, etc., and thus be parallelised trivially.

We should briefly clear here a possible language ambiguity about the terms local and global. By *local* and *global* minimisation, we mean algorithms capable of finding local minima of some function, or of finding the minimum of a function over the whole domain, respectively. It should be stressed that the algorithms used in Radio Astronomy are local minimisers, thus solutions may depend to some extent on starting points (depending on the convexity of the problem).

The term *global solver* is commonly used in Radio Astronomy to indicate coupled algorithms capable of solving for multiple/all frequencies and time slots. These would require more complex methods, increased computational costs and complex parallelism, and will be briefly discussed in section 4.

The minimisations required are non-linear, requiring iterative solutions irrespective of the techniques used. In general, it is not possible to determine in advance the number of iterations required to achieve convergence to within a given tolerance: it depends on a host of issues, such as the algorithm, the actual problem and its properties, the initial guess, etc. Lower numbers of iterations could be required if good starting values were available: for example, if neighbouring frequencies needed calibrating and the results for one frequency were used as starting values for the next one; likewise for multiple snapshots.

2.3.1 The Levenberg-Marquardt Algorithm

The Levenberg-Marquardt Algorithm (LMA) is a canonical method for the solution of non-linear Least Squares problems (RD04). LMA requires for each iteration step to compute the products of the Jacobian J of the elements of the least squares elements (basically, the number of points in the visibility) wrt. the parameters: $J^\dagger J$ (an approximate Hessian) and $J^\dagger x$ where x is a vector. Note that for a visibility matrix of size $N_{\rm vis}$ with full polarisation, J is a $2N_{\rm vis} \times N_{\rm vis}^2$ complex matrix and x is a vector of length $N_{\rm vis}^2$. Each iteration requires also the solution of a system of equations.

In the case of gain computation, because of the structure of J, it should be possible to reduce the number of operations required. Table 6 shows the number of operations required for each iteration. Standard counts are used for the solver. Also, the memory requirements are shown: unfortunately, it is not possible to be precise as they may depend on the problem being solved.

	Full LMA	**LMA using shortcut**
Compute J	$\mathcal{O}(N_{\rm it}N_{\rm rec}^3)$	$\mathcal{O}(N_{\rm it}N_{\rm vis}^3)$
$J^\dagger J$	$32N_{\rm it}N_{\rm vis}^4$	$\mathcal{O}(N_{\rm it}N_{\rm vis}^3)$
$J^\dagger X$	$32N_{\rm it}N_{\rm vis}^3$	$\mathcal{O}(N_{\rm it}N_{\rm vis}^3)$
Solver	$\mathcal{O}(N_{\rm it}N_{\rm vis}^3)$	$\mathcal{O}(N_{\rm it}N_{\rm vis}^3)$
Overall	$32N_{\rm it}N_{\rm vis}^4 + \mathcal{O}(N_{\rm it}N_{\rm vis}^2)$	$\mathcal{O}(N_{\rm it}N_{\rm vis}^3)$
Memory (complex)	$\mathcal{O}(N_{\rm it}N_{\rm vis}^3)$	$\mathcal{O}(N_{\rm it}N_{\rm vis}^2)$

Table 6: Levenberg-Marquardt Algorithm: computational complexity.

It is not possible to define a more precise operation count for the various components of the LMA, apart from the explicit computation of the approximate Hessian from the Jacobian.

Some operation counts for the full case can be given (ignoring any symmetry) corresponding to dense matrix-matrix products (aka ZGEMM), matrix-vector products (aka ZGEMV) and a dense linear solver ($\mathcal{O}(N_{\rm vis}^3)$).

In general, iterative methods are not useful for dense problems. However, if a very good approximation to the solution is available (for example from a previous frequency/snapshot), an iterative solver could be used at the cost of $\mathcal{O}(N_{\rm vis}^2)$ operations per iteration.

The various steps of the LMA can be multithreaded, although efficiency very much depends on the problem size (smaller problems leading to less efficiency). However, as operations are dominated by matrix-matrix operations, efficiency is expected to be reasonably high. It should be noted that LMA can have a considerable memory footprint. Distributed LMA (e.g. using MPI) can only become efficient when the amount of local computation is very large with respect to the communication costs, and load balancing is very accurate, hence for large problems.

The flops count shows that while LMA can be used for instruments consisting of small numbers of receivers, its computational costs increase far too quickly with the problem size to be of practical use for the larger instruments currently under development.

2.3.2 StEFCal

StEFCal (Statistically Efficient and Fast Calibration) is an alternating direction implicit (ADI) method for the calibration of visibilities for both the scalar (RD05) and fully polarised case (RD06).

The numerical complexity of StEFCal, its properties and its range of applicability are reported in the references cited above as well as in Table 7.

Scalar (no polarisation)	Full polarisation
$24\,N_{\rm it}\,N_{\rm rec}^2$	$48\,N_{\rm it}\,N_{\rm rec}^2\,N_{\rm pol}$

Table 7: StEFCal: operation count.

Where $N_{\rm it}$ is the number of iterations (in general, $N_{\rm it}\leq$50, even smaller for non-polarised cases), depending on the convergence required. The approximate number of iterations should be viewed as just a ball-park figure, based on using StEFCal within MeqTrees, in calibrating LOFAR data, and with numerical simulations, etc., and if anything, it should be viewed as pessimistic.

The great difference in the operation count between LMA and StEFCal for the same problem domain is algorithmic and does not depend on any specific data properties.

It should be noted that all data are accessed linearly (contiguous locations) hence StEFCal can achieve very high performance levels. As the computation is extremely fast, StEFCal could be executed on standard CPUs (for example on the host system for accelerators such as GPUs) as computation would be much faster than other required steps, such as gridding or FFTs.

StEFCal is used successfully in LOFAR and within MeqTrees where it has proved to be not only much faster than existing solvers (by orders of magnitude) but also numerically comparable to existing solvers and more flexible.

In general (unless the problem is so small as to fit in cache), each iteration requires access to the observed as well as the model sky visibility matrices ($2N_{\rm rec}^2N_{\rm pol}$ complex numbers). The number of operations per data point loaded is therefore low. However, all data access is through contiguous locations of memory, thus maximising the use of loaded cache lines as well as making best use of prefetching thus enhancing measured performance: on current CPUs on a single core measured performance reaches up to 35% of peak with near perfect scalability

with size (RD05, RD06). The memory footprint is also very small, about $2N_{\rm rec}^2 N_{\rm pol} + \mathcal{O}(N_{\rm rec})$ complex numbers (words), corresponding to the initial visibility matrices.

Because of its very small memory footprint, its speed and operation count, we expect that very sizeable problems could be run on individual systems (or computational engines within a system). It should be noted that StEFCal offers also the possibility of internal (i.e. within the algorithm) multi-threading.

Implementations on both CPUs and GPUs have been carried out. However, given the high speed of the algorithm, it could be envisaged that StEFCal could be performed on the host (CPU) thus leaving any accelerator (such as GPUs or other) available for more demanding computations. StEFCal efficiency is lower on GPUs than on CPUs, because of its relatively low ratio of flops per data access.

Direction Dependent Effects (DDE) can be included in StEFCal, as discussed in section 3.

2.4 Subtract (remove known visibilities from observed visibilities)

Subtraction of the corrupted model visibilities V^M from the observed visibilities require the application of eq. (4). This is very inexpensive and the costs, computed using Table 3, are summarised in Table 8.

Operation	Total nr. of flops
Corrupt: $W = \mathcal{G}\, V^M\, \mathcal{G}^\dagger$	$\approx 64 N_{\rm rec}^2$
Subtract: $V^D = V - W$	$\approx 8\, N_{\rm rec}^2$
Product, then product and subtract: $W = \mathcal{G}\, V^M$, then $V^D = V - W\, \mathcal{G}^\dagger$	$\approx 64\, N_{\rm rec}^2$

Table 8: Subtraction step: operations count.

The table also shows that it is possible to combine a matrix multiplication with the subtraction to reduce the number of operations.

Naturally, subtractions are computed only for half the visibility matrix.

2.5 Correct (correct the updated observed visibilities)

The last step consists of correcting the updated observed visibilities as in eq. (5). This consists of computing the inverse of each diagonal block of the complex gains matrix $\mathcal{G}$ then applying the inverse gains to both sides of the updated observed visibilities (see Table 9).

Operation	Nr. of Operations
compute $\mathcal{G}^{-1}$	$\mathcal{O}(N_{\rm rec})$
Compute $V^C = \mathcal{G}^{-1} V^D\, \mathcal{G}^{-\dagger}$	$64\, N_{\rm rec}^2$

Table 9: Correct: operation count

Again, only half of the corrected visibility needs to be computed.

2.6 Approximate Total Costs per Observation per Second

As we do not have as yet further information, we assume a worst case scenario (one calibration for each cross-correlation dump with dump time $t_{\rm dump}$).

We assume that:

- Calibration is carried out for each of the $N_{\rm ch}$ channels and each of the $N_{\rm beam}$ beams.
- We need $N_{\rm major}$ CLEAN major and $N_{\rm SelfCal}$ SelfCal cycles;
- Calibration is required for each CLEAN major cycle and each SelfCal cycle, under the assumption that they are nested;
- A calibration is carried out for each dump time $t_{\rm dump}$;
- Given an observation time $t_{\rm obs}$ we allocate the same time to process the data;
- We are interested in the required computational rate, (FLOPS = flops/second); hence for the previous two points the observation time disappears from the computational rates;
- $N_{\rm source}$ sources need to be removed;
- Some compression rate c_r (BDA suggests $c_r \geq 5$, possibly much greater) could be applied.

Hence we summarise the leading terms contributing to the costs, in terms of required, deliverable FLOPS, where we have extracted the common factor $\mathcal{F}$, in Table 2.

Common factor $\mathcal{F}$	$\approx \frac{64\, N_{\rm beam}\, N_{\rm ch}\, N_{\rm SelfCal}\, N_{\rm major} N_{\rm rec}^2}{c_r\, t_{\rm dump}}$
Operation	**FLOPS**
Predict	$64\, N_{\rm source}\, \mathcal{F}$
Solve	$48\, N_{\rm it}\, N_{\rm pol}\, \mathcal{F}$
Subtract	$64\, \mathcal{F}$
Correct	$64\, \mathcal{F}$
Total	$[64\,(N_{\rm source}+2) + 48\, N_{\rm it}\, N_{\rm pol}]\, \mathcal{F}$

Table 10: Overall calibration computational rate required: worst case scenario

3 Direction Dependent Calibration

This section introduces the issue of the calibration of Direction Dependent Effects (DDE). This will have an impact on the scientific output from the SKA1. However, at the time of writing, the field is not yet mature enough to warrant any more than a qualitative discussion, but it is the subject of active studies.

DDEs could be introduced in the calibration process in one of two ways:

1. Optimise all directions at the same time. This would require $\mathcal{O}(N_{\rm dir}^2)$ operations. This approach is taken in RD03and RD11.

2. Subtract as well as possible the contributions from all directions except that of interest, and then optimise for this direction with direction independent calibration. Iterate over the number of directions. Possibly, more iterations over all directions of interest are required. This method has been named *peeling* (RD12), adapted as differential gains (RD10), and another variant is currently successfully being applied to LOFAR HBA data under the name Facet Calibration (RD08).

Note that the direction dependent gains obtained in both methods cannot be applied to visibility data (as in section 2) without passing through the image plane, requiring either an imager capable of imaging direction dependent gains (e.g. the LOFAR AWImager) or imaging the sky facet by facet.

Two actual algorithms are briefly discussed below: SAGECal (RD03, RD07) and StEFCal for DDE.

3.1 SAGECal

From a signal processing point of view, calibration is a Maximum Likelihood (ML) estimation of the sky and instrument parameters. Traditional calibration techniques, such as LMA and StEFCal estimate the ML by solving a non-linear Least Squares (LS).

For the SKA, given the unprecedented number of receivers that must be calibrated for multiple directions in the sky, speed of convergence is of paramount importance. That led S. Yattawatta and S. Kazemi (RD03, RD07) towards the use of Expectation Maximisation (EM) type algorithms.

EM algorithms iteratively estimate ML and benefit from the essential property of increasing the likelihood at every iteration. Therefore, if they also converge fast, they could be one of the best candidates to be applied to the calibration problem. The Space Alternating Generalised Expectation Maximisation (SAGE) technique is one of these candidates which can be much faster than the LS calibration method. Initially introduced by (RD02) in the context of signal processing, the method has been applied to radio interferometric calibration problem (RD03, RD07).

The key point in the high speed of the SAGE method is that it can solve the ML estimation problem over different individual directions in the sky, while at each iteration, it estimates solutions using the latest updates obtained from the previous iteration. In other words, calibrating N_{rec} receivers using N_{pol} polarisations for N_{dir} directions in the sky, the SAGE method breaks the ML estimation problem into N_{dir} sub-ML estimation problems. Its computational complexity therefore is $\mathcal{O}(N_{\mathrm{it}}N_{\mathrm{dir}}N_{\mathrm{vis}}^2)$. Computing for all directions simultaneously would require $\mathcal{O}(N_{\mathrm{it}}(N_{\mathrm{dir}}N_{\mathrm{vis}})^2)$ operations.

However, no more detailed information is currently available about the number of operations or the number of iterations required.

3.2 StEFCal for Direction Dependent Calibration

StEFCal could be used in either of two ways:

- All directions are calibrated at the same time.
- One direction is “peeled” before proceeding with the next direction

In the first case, for the computational costs, instead of $N_{\rm vis}$ one should use $N_{\rm dir} \cdot N_{\rm vis}$ where $N_{\rm dir}$ is the number of directions and $N_{\rm vis} = N_{\rm rec} \cdot N_{\rm pol}$. Directions are, in this case, obviously coupled. The second case corresponds to uncoupled directions (i.e. gains in one direction do not affect gains in another direction). Basically, one computes the gains along a direction, then the calibrated sky is removed and another direction is tackled, etc. It may be the case that each direction has to be computed and removed more than once. However, computational costs in the tables depend linearly on the factor $N_{\rm dir}$.

O. Smirnov has included StEFCal for DDE computation within MeqTrees using the second approach to good effect.

4 “Global” Solvers

By global solvers here we mean optimisations that require multiple frequencies (channels) and/or times (snapshot): in other words, there is a direct or indirect coupling between multiple snapshots, frequencies, directions, etc.

We should stress that when the number of frequencies and snapshots is limited (such as, for example, when the uncoupled problems would be solved in the same computing island, thus requiring a limited number of computational engines), then the problems could be expected to be solved reasonably easily. This could be, for example, the case of smoothing, for example polynomial smoothing, over a number of snapshots and/or frequencies.

Difficulties increase with problem sizes and the solution process needs to straddle across multiple computing islands, thus requiring many computational engines. In such cases, distributed parallel algorithms would be required.

LOFAR Black-Board Self Calibration (BBS) currently has a global solver, but it is very rarely used. Also, while CASA has no such facilities, it could be possible to create a global solver from CASACORE components.

The methods in the previous section probably cannot cope so well in highly distributed situations, and there is little incentive to study methods and provide estimates of the number of operations required when clear needs have not as yet been defined.

It should be noted that S. Yatawatta (ASTRON) has announced distributed SAGECal, based on Consensus Optimisation (RD01) that could be used for large, distributed problems. This is being studied and applied to LOFAR by S. Yatawatta (RD07) – obviously, the SKA will need to follow closely its development.

It should also be noted that versions of StEFCal that allow multi-snapshot and/or multi-channel smoothing (including polynomial smoothing) have been studied and been implemented in an experimental way to good effect.

In any case, the need for large-scale coupled solvers needs to be part of the SDP risk register. Further analysis should be carried out post-PDR.

5 Extended Sources

Unresolved point sources are the ideal candidates when building a source model for use in calibration. Moderately extended sources could also be included if they have sufficient signal to noise ratios and have a source structure that can be modelled accurately. Indeed, this could be the case with SKA1, particularly for some science aims requiring at the accurate removal of sources.

In theory, extended objects could be represented by sets of delta-functions. Given the resolution of the instrument, very large numbers of delta functions would be required, thus leading to considerable, perhaps unacceptable, computational costs; also, any attempt to reduce the size of these sets could lead to modelling errors.

A better approach could be to decompose extended sources into 2D components. Ideally these would be functions with a "simple", possibly analytic representation in Fourier space for a direct representation of the extended objects in u, v, w space. Also, these functions could allow an "economical" expansion of complicated objects over a limited set of functions, without jeopardising computational costs.

Multi-Scale Clean (RD13), which models extended emission as a sets of scaled truncated inverted paraboloids, multiplied by a first order spheroidal function, is currently the most commonly adopted approach. Yatawatta has investigated the decomposition into sets of orthonormal basis functions such as shapelets (RD14).

As already mentioned in subsection 2.2, the model sky visibilities corresponding to an extended source can be obtained from the FFT on an image, followed by degridding. The cost of this procedure is independent of the number of sources, but depends on the number of pixels ($N_{\rm pix}^2$) of the model image as $5N_{\rm pix}^2 \log_2(N_{\rm pix})$ + degridding cost.

Extended sources are currently being actively studied, but the field is not sufficiently mature to allow an in-depth study of it, yet alone reporting on computational costs. The topic has been included here as an item for future investigation as it could play an important role in the development of the SKA1.

6 Data and Metadata

Making a distinction between data and metadata is difficult and, to some extent, subjective. Here, the distinction is between items of data moving synchronously and at the same rate as the main data flow (visibilities), and items that need be made available but can be delivered asynchronously.

Data and metadata are reported here as items that need to be made available to the calibration as a whole, rather than to individual components. Further specifications are beyond the scope of this Memo.

We do not specifically describe the input and output to the various calibration stages. Following the graphic approach described in the Pipeline Design document (AD02), data would be encapsulated in data *drops*, and it is envisaged that functional drops would be connected to them to process the data, without moving them.

It should also be stressed that these tables are based on, as yet, incomplete information: a calibration strategy has not been released by the SKAO at the time of writing; a detailed data design for the calibration stages is not available either. As a consequence, the information here, in particular that included in the tables, are to be seen as an initial approach, which will be refined, extended and amended in due course.

We will assume a worst case scenario: all data are in double precision (1 real = 8 bytes; 1 complex = 16 bytes) and a calibration per correlation dump. As said in the Introduction (section 1) DDE will not be taken into account, but only Direction Independent calibration.

6.1 Data

The data required for the calibration are listed in Table 11. Complex gains are sent to the Local Telescope State (LTS), unless their data volume gets too large (as is the case for the Self-Calibration pipeline). In such cases the gains will be treated as a Data Product and use the Drop channel interface, see AD02.

Item	From	To	Refresh	Size (Bytes)
Visibilities	CSP		Dump ($t_{\rm dump}$)	$16\,N_{\rm beam}\,N_{\rm ch}\,N_{\rm pol}\,N_{\rm rec}^2/2$
u, v, w coordinates (metres)	Receive Function		Dump ($t_{\rm dump}$)	$24\,N_{\rm beam}\,N_{\rm rec}^2/2$
Time stamp	Receive Function		Dump ($t_{\rm dump}$)	$\mathcal{O}(1)$
Complex gains G_a		LTS	Dump ($t_{\rm dump}$)	16 $N_{\rm ch}N_{\rm pol}N_{\rm rec}$
Failure/success/flagging/ diagnostics etc.		LTS	Dump ($t_{\rm dump}$)	TBD

Table 11: The data required for the calibration.

6.2 Metadata

We have divided the metadata in a number of categories (see Table 12):

- Instrument model
- Sky model
- Atmosphere model
- Observation data and parameters
- Computational Parameters

Item	Source	Refresh	Size
Instrument Model			
Station beam data Tables? Parameterised model?	LTS	TBD (Almost static ?)	$\mathcal{O}(N_{\mathrm{beam}})$
Band-pass data	LTS	Almost static	TBD
Sky Model			
Global Sky Model	TM LSM	TBD (Almost static ?)	$\mathcal{O}(N_{\mathrm{source}})$
Local Sky Model	LSM	TBD	$\mathcal{O}(N_{\mathrm{source}})$
Atmosphere Model			
Ionosphere model and parameters	TBD	TBD	$\mathcal{O}(N_{\mathrm{source}})$
Troposphere?	LTS	TBD	$\mathcal{O}(N_{\mathrm{source}})$
Observation Data and Parameters			
Channels (Frequencies)	LTS	Static	$\mathcal{O}(N_{\mathrm{ch}})$
Cross-correlation dump time (t_{dump})	LTS	Static?	1
Computational Parameters			
Algorithmic: choice and parameters (max. iterations, convergence, etc.) It may depend on sub-band, etc.	Execution Framework	Static	Depends on the algorithms selected. E.g. StEFCal requires 5 or 6 numbers.

Table 12: Metadata required for the calibration.

7 Examples

Possible values for the SKA1-LOW and MID are listed in Table 13.

Symbol	Description	LOW	MID
N_{rec}	Nr. of receivers	512	197
N_{pol}	Nr. of polarisations	4	4
$N_{\mathrm{beam}} \times N_{\mathrm{ch}}$	Total number of beam/channels	65536	65536
t_{dump}	Dump time	0.6 sec	0.14 sec
N_{source}	Nr. of sources in model sky	200	200
N_{it}	Nr. of iterations for StEFCal	20	20
$N_{\mathrm{SelfCal}} \times N_{\mathrm{major}}$	total number of calibrations per sample	20	20
c_r	Compression rate	5	5

Table 13: Possible parameter values for SKA1-LOW and MID.

Costs are only computed for known items. The following are the limitations of the model:

- All data are assumed to be in double precision (16 bytes per complex number)
- Extended sources are not considered
- Solvers other than StEFCal are not considered
- Generation of Jones Matrices (apart from geometric phase factors) is not considered
- Direction-dependent calibration is not considered
- Only two-sided application of Jones matrices is considered.

Using Tables 10 and 13 we can then assemble approximate computational rates figures for the calibration for an observation in terms of FLOPS to be *delivered* (flops/second). These are reported in Table 14.

Operation	TFLOPS	
	LOW	MID
Predict	1470	930
Solve	440	280
Subtract	8	5
Correct	8	5
Total	1920	1220

Table 14: Computational costs for SKA1-LOW and MID.

The bandwidth required for the visibilities from the cross-correlator to the SDP engine, based on the tables in section 6.1 is calculated in Table 15 *without* including the compression rate c_r.

Instrument	Bandwidth (GBytes/sec)
LOW	920
MID	580

Table 15: Bandwidth required from the CSP cross-correlator to the SDP.

A Relationship Between Jones Matrix and Mueller Formalism

The application of Jones matrices to the brightness matrix B_s was discussed in subsection 2.1.

Given the complex matrices A, B and X, the following equality holds:

$$A\,X\,B^{\dagger} = (\mathrm{conj}(B) \otimes A) \cdot \mathrm{vec}(X) \tag{11}$$

where $\mathrm{conj}(B)$ is the complex conjugate (*without* transposition) of B; $\mathrm{vec}(X)$ stacks one above the other in a single column vectors the columns of X, namely:

$$\mathrm{vec}(X) = \begin{pmatrix} X_{1,1} \\ \vdots \\ X_{N,1} \\ \vdots \\ X_{1,N} \\ \vdots \\ X_{N,N} \end{pmatrix}$$

The symbol $\otimes$ denotes the Kronecker product of two matrices which results in a matrix having dimensions equal to the product of the dimensions of the matrices, namely:

$$\mathrm{conj}(B) \otimes A = \begin{pmatrix} A_{1,1}\,\mathrm{conj}(B) & \cdots & A_{1,N}\,\mathrm{conj}(B) \\ \vdots & \ddots & \vdots \\ A_{N,1}\,\mathrm{conj}(B) & \cdots & A_{N,N}\,\mathrm{conj}(B) \end{pmatrix}$$

we can now apply eq. (11) to the generic Jones Matrix $W_{a,s}$ where a is the receiver index and s the source index:

$$W_{a,s}\,B_s\,W_{b,s}^{\dagger} = (\mathrm{conj}(W_{b,s}) \otimes W_{a,s}) \cdot \mathrm{vec}(B_s) = M_{a,b,s} \cdot b_s^M \tag{12}$$

where $M_{a,b,s}$ is the 4x4 Mueller matrix and b_s^M the Mueller vector.

The double receiver index on the Mueller matrices show that they depend on receiver pairs, not on single receivers. A Mueller matrix needs to be generated for each pair of receivers. Taking into account the Hermitian properties of the visibilities (so that only half the elements of the visibility matrix are computed) the computational costs are summarised in Table 16.

Operation	**Total nr. of flops**	
	Full Jones Matrix	Diagonal Jones Matrix
Generate all $M_{a,b,s}$	$64\,N_{\mathrm{rec}}^2\,N_{\mathrm{source}}$	$32\,N_{\mathrm{rec}}^2\,N_{\mathrm{source}}$
Apply all $M_{a,b,s}$	$64\,N_{\mathrm{rec}}^2\,N_{\mathrm{source}}$	$32\,N_{\mathrm{rec}}^2\,N_{\mathrm{source}}$
Total	$128\,N_{\mathrm{rec}}^2\,N_{\mathrm{source}}$	$64\,N_{\mathrm{rec}}^2\,N_{\mathrm{source}}$

Table 16: Mueller formalism computational costs

Table 16 shows that the computational costs are twice as high as for the Jones matrix formalism and four times as high as for the one-sided application of Jones matrices (cfr. subsection 2.1 and Table 3).