Mon. Not. R. Astron. Soc. **000**, 1–14 (2014) Printed 22 October 2014 (MN LATEX style file v2.2)

Radio interferometeric calibration as a complex optimization problem

N. Bourbaki[1⋆]
[1] *Institute For The Advancement Of Complex Chicken Recipes*

in original form 2014 February 2

ABSTRACT
Recent developments in optimization theory have extended some traditional algorithms for least-squares optimization of real-valued functions (Gauss-Newton, Levenberg-Marquardt, etc.) into the domain of complex functions of a complex variable. This employs a formalism called the Wirtinger derivative, and derives a full-complex Jacobian counterpart to the conventional real Jacobian. We apply these developments to the problem of radio interferometric gain calibration, and show how the general complex Jacobian formalism, when combined with conventional optimization approaches, yields a whole new family of calibration algorithms, including those for the polarized and direction-dependent gain regime. We further extend the Wirtinger calculus to an operator-based matrix calculus for describing the polarized calibration regime. Using approximate matrix inversion results in computationally efficient implementations; we show that some recently proposed calibration algorithms such as StefCal and peeling can be understood as special cases of this, and place them in the context of the general formalism. Finally, we present an implementation and some applied results of CohJones, another specialized direction-dependent calibration algorithm derived from the formalism.

Key words: Instrumentation: interferometers, Methods: analytical, Methods: numerical, Techniques: interferometric

1 INTRODUCTION

Chicken chicken chicken. Chicken chicken? Chicken[1] chicken, chicken (Chicken & Chicken 2014).

2 PRIOR WORK

2.1 Complex optimization & Wirtinger calculus

The traditional approach to optimizing a function of n complex variables $f(\boldsymbol{z})$, $\boldsymbol{z} \in \mathbb{C}^n$ is to treat the real and imaginary parts $\boldsymbol{z} = \boldsymbol{x} + i\boldsymbol{y}$ independently, turning f into a function of $2n$ real variables $f(\boldsymbol{x}, \boldsymbol{y})$, and the problem into an optimization over $\mathbb{R}^{2n}$.

Laurent et al. (2012) have developed an alternative approach to the problem based on Wirtinger calculus. The central idea of Wirtinger calculus is to treat $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$ as independent variables, and optimize $f(\boldsymbol{z}, \bar{\boldsymbol{z}})$ using the Wirtinger derivatives

$$\frac{\partial}{\partial z} = \frac{1}{2}\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right), \quad \frac{\partial}{\partial \bar{z}} = \frac{1}{2}\left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right),$$

where $z = x + iy$. It is easy to see that

$$\frac{\partial \bar{z}}{\partial z} = \frac{\partial z}{\partial \bar{z}} = 0,$$

i.e. that $\bar{z}$ (z) is treated as constant when taking the derivative with respect to z $(\bar{z})$. From this it is straightforward to define the *complex gradient* operator

$$\frac{\partial}{\partial^C \boldsymbol{z}} = \left[\frac{\partial}{\partial \boldsymbol{z}} \frac{\partial}{\partial \bar{\boldsymbol{z}}}\right] = \left[\frac{\partial}{\partial z_1} \cdots \frac{\partial}{\partial z_n} \frac{\partial}{\partial \bar{z}_1} \cdots \frac{\partial}{\partial \bar{z}_n}\right],$$

from which definitions of the complex Jacobian and complex Hessians naturally follow. The authors then show that various optimization techniques developed for real functions can be reformulated using complex Jacobians and Hessians, and applied to the complex optimization problem. Of particular interest to us, they generalize the Gauss-Newton (GN) and Levenberg-Marquardt (LM) methods for solving the non-linear least squares (LS) problem

$$\min_{\boldsymbol{z}} ||\boldsymbol{r}(\boldsymbol{z}, \bar{\boldsymbol{z}})||_F, \tag{1}$$

⋆ E-mail: nbourbaki@pantheon.fr
[1] Chicken chicken!

where $\boldsymbol{r}$ has values in $\mathbb{C}^m$, and $||\cdot||_F$ is the Frobenius norm. This is done as follows. Let us formally treat $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$ as independent variables, define an *augmented parameter vector* containing both,

$$\check{\boldsymbol{z}} = \begin{bmatrix}\boldsymbol{z}\\ \bar{\boldsymbol{z}}\end{bmatrix}$$

and designate its value at step k by $\check{\boldsymbol{z}}_k$. Then, define

$$\boldsymbol{J}_k = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{z}}(\check{\boldsymbol{z}}_k),\ \boldsymbol{J}_{\bar{k}} = \frac{\partial \boldsymbol{r}}{\partial \bar{\boldsymbol{z}}}(\check{\boldsymbol{z}}_k),\ \check{\boldsymbol{r}}_k = \begin{bmatrix}\boldsymbol{r}(\check{\boldsymbol{z}}_k)\\ \bar{\boldsymbol{r}}(\check{\boldsymbol{z}}_k)\end{bmatrix} \tag{2}$$

We'll call the $m \times n$ matrices $\boldsymbol{J}_k$ and $\boldsymbol{J}_{\bar{k}}$ the *partial* and *partial conjugate Jacobian*, respectively, and the $2m$-vector $\check{\boldsymbol{r}}_k$ the *augmented residual vector*. The *complex Jacobian* of the cost function $||\check{\boldsymbol{r}}(\check{\boldsymbol{z}})||_F$ can then be written (in block matrix form) as

$$\boldsymbol{J} = \begin{bmatrix} J_k & J_{\bar{k}} \\ \bar{J}_{\bar{k}} & \bar{J}_k \end{bmatrix}, \tag{3}$$

Note that this is a $2m \times 2n$ matrix. The LM parameter update step is then defined as

$$\delta\check{\boldsymbol{z}} = \begin{bmatrix}\delta\boldsymbol{z}\\ \delta\bar{\boldsymbol{z}}\end{bmatrix} = -(\boldsymbol{J}^H\boldsymbol{J} + \lambda\mathbb{I})^{-1}\boldsymbol{J}^H\check{\boldsymbol{r}}_k, \tag{4}$$

where $\mathbb{I}$ is the identity matrix, and λ is the LM damping parameter. When $\lambda = 0$ this becomes equivalent to the Gauss-Newton (GN) method; with $\lambda \to \infty$ this corresponds to steepest descent (SD) with ever smaller steps.

An alternative version of LM is formulated as

$$\delta\check{\boldsymbol{z}} = -(\boldsymbol{J}^H\boldsymbol{J} + \lambda\boldsymbol{D})^{-1}\boldsymbol{J}^H\check{\boldsymbol{r}}_k, \tag{5}$$

where $\boldsymbol{D}$ is simply the diagonalized version of $\boldsymbol{J}^H\boldsymbol{J}$.

Note that while $\delta\boldsymbol{z}$ and $\delta\bar{\boldsymbol{z}}$ are formally computed independently, the structure of the equations ensures that the results are consistent, i.e. that $\overline{\delta\boldsymbol{z}} = \delta\bar{\boldsymbol{z}}$. In practice this redundancy usually means that only half the calculations need to be performed.

Laurent et al. (2012) show that Eq. 4 yields exactly the same system of LS equations as would have been produced had we treated $\boldsymbol{r}(\boldsymbol{z})$ as a function of real and imaginary parts $\boldsymbol{r}(\boldsymbol{x}, \boldsymbol{y})$, and taken ordinary derivatives in $\mathbb{R}^{2n}$. However, the complex Jacobian may be easier and more elegant to derive analytically, as we'll see below in the case of radio interferometric calibration.

3 UNPOLARIZED CALIBRATION

In this section we will apply the formalism above to the simpler case of unpolarized calibration. This will then be extended to the fully polarized case in Sect. 5.

3.1 Direction-independent calibration

Let us first explore the simplest case of direction-independent (DI) calibration. Consider an interferometer array of $N_{\rm ant}$ antennas measuring $N_{\rm bl} = N_{\rm ant}(N_{\rm ant}-1)/2$ pairwise visibilities. Each antenna pair pq ($1 \leqslant p < q \leqslant N_{\rm ant}$) measures the visibility

$$g_p m_{pq} \bar{g}_q + n_{pq}, \tag{6}$$

where m_{pq} is the (assumed known) sky coherency, g_p is the (unknown) complex gain parameter associated with antenna p, and n_{pq} is a complex noise term that is Gaussian with a mean of 0 in the real and imaginary parts. The calibration problem then consists of estimating the complex antenna gains $\boldsymbol{g}$ by minimizing residuals in the LS sense:

$$\min_{\boldsymbol{g}} \sum_{pq} |r_{pq}|^2, \quad r_{pq} = d_{pq} - g_p m_{pq} \bar{g}_q, \tag{7}$$

where d_{pq} are the observed visibilities. Treating this as a complex optimization problem as per the above, let us write out the complex Jacobian. With a vector of $N_{\rm ant}$ complex parameters $\boldsymbol{g}$ and $N_{\rm bl}$ measurements d_{pq}, we'll have a full complex Jacobian of shape $2N_{\rm bl} \times 2N_{\rm ant}$. It is conventional to think of visibilities laid out in a visibility matrix; the normal approach at this stage is to vectorize d_{pq} by fixing a numbering convention so as to enumerate all the possible antenna pairs pq ($p < q$) using numbers from 1 to $N_{\rm bl}$. Instead, let us keep using pq as a single "compound index", with the implicit understanding that pq in subscript corresponds to a single index from 1 to $N_{\rm bl}$ using *some* fixed enumeration convention. Where necessary, we'll write pq in square brackets (e.g. $a_{[pq],i}$) to emphasize this.

Now consider the corresponding partial Jacobian $\boldsymbol{J}_k$ matrix (Eq. 2). This is of shape $N_{\rm bl} \times N_{\rm ant}$. We can write the partial Jacobian in terms of its value at row $[pq]$ and column j as

$$[\boldsymbol{J}_k]_{[pq],j} = \begin{cases} -m_{pq}\bar{g}_q, & j = p, \\ 0, & \text{otherwise.} \end{cases}$$

In other words, within each column j, J_k is only nonzero at rows corresponding to baselines $[jq]$. We can express this more compactly using the Kronecker delta:

$$J_k = -\overbrace{\left[m_{pq}\bar{g}_q\delta_p^j\right]}^{j=1\ldots N_{\rm ant}}\Big\}_{[pq]=1\ldots N_{\rm bl}\ (p<q)v} \tag{8}$$

Likewise, the conjugate partial Jacobian $\boldsymbol{J}_{\bar{k}}$ may be written as

$$J_{\bar{k}} = -\overbrace{\left[g_p m_{pq}\delta_q^j\right]}^{j=1\ldots N_{\rm ant}}\Big\}_{[pq]=1\ldots N_{\rm bl}\ (p<q)} \tag{9}$$

A specific example is provided in Appendix A. The full complex Jacobian (Eq. 3) then becomes, in block matrix notation,

$$\boldsymbol{J} = -\begin{bmatrix} \overbrace{m_{pq}\bar{g}_q\delta_p^j}^{j=1\ldots N_{\rm ant}} & \overbrace{g_p m_{pq}\delta_q^j}^{j=1\ldots N_{\rm ant}} \\ \bar{m}_{pq}\bar{g}_p\delta_q^j & g_q\bar{m}_{pq}\delta_p^j \end{bmatrix} \begin{matrix} \Big\}_{[pq]=1\ldots N_{\rm bl}}\ (p<q) \\ \Big\}_{[pq]=1\ldots N_{\rm bl}}\ (p<q) \end{matrix} \tag{10}$$

where the $[pq]$ ($p < q$) and j subscripts within each block span the full range of $1 \ldots N_{\rm bl}$ and $1 \ldots N_{\rm ant}$. Now,

since $d_{pq} = \bar{d}_{qp}$ and $m_{pq} = \bar{m}_{qp}$, we may notice that the bottom half of the augumented residuals vector $\breve{\boldsymbol{r}}$ corresponds to the conjugate baselines qp $(q > p)$:

$$\breve{\boldsymbol{r}} = \begin{bmatrix} r_{pq} \\ \bar{r}_{pq} \end{bmatrix} = \begin{bmatrix} d_{pq} - g_p m_{pq} \bar{g}_q \\ \bar{d}_{pq} - \bar{g}_p \bar{m}_{pq} g_q \end{bmatrix} = \begin{bmatrix} r_{pq} \\ r_{qp} \end{bmatrix} \begin{matrix} \}_{[pq]=1\ldots N_{\rm bl}\ (p<q)} \\ \}_{[pq]=1\ldots N_{\rm bl}\ (p<q)} \end{matrix}$$

as does the bottom half of $\boldsymbol{J}$ in Eq. 10. Note that we are free to reorder the rows of $\boldsymbol{J}$ and $\breve{\boldsymbol{r}}$ and intermix the normal and conjugate baselines, as this will not affect the LS equations derived at Eq. 4. This proves most convenient: instead of splitting $\boldsymbol{J}$ and $\breve{\boldsymbol{r}}$ into two vertical blocks with the compound index $[pq]$ $(p < q)$ running through $N_{\rm bl}$ rows within each block, we can treat the two blocks as one, with a single compound index $[pq]$ $(p \neq q)$ running through $2N_{\rm bl}$ rows:

$$\boldsymbol{J} = -\Big[\overbrace{m_{pq}\bar{g}_q \delta_p^j}^{j=1\ldots N_{\rm ant}} \quad \overbrace{g_p m_{pq} \delta_q^j}^{j=1\ldots N_{\rm ant}} \Big],\ \breve{\boldsymbol{r}} = \begin{bmatrix} r_{pq} \end{bmatrix} \ \}_{[pq]=1\ldots 2N_{\rm bl}} \tag{11}$$

where for $q > p$, $r_{pq} = \bar{r}_{qp}$ and $m_{pq} = \bar{m}_{qp}$. For clarity, we may adopt the following order for enumerating the row index $[pq]$: $12, 13, \ldots, 1n,\ 21, 22, \ldots, 2n,\ 31, 32, \ldots, 3n,$ $\ldots, n1, \ldots, n\,n-1$.

Equation A1 in the Appendix provides an example of $\boldsymbol{J}$ for the 3-antenna case. For brevity, let us define the shorthand

$$y_{pq} = m_{pq}\bar{g}_q.$$

We can now write out the structure of $\boldsymbol{J}^H\boldsymbol{J}$. This is Hermitian, consisting of four $N_{\rm ant} \times N_{\rm ant}$ blocks:

$$\boldsymbol{J}^H\boldsymbol{J} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^H & \boldsymbol{A} \end{bmatrix} \tag{12}$$

since the value at row i, column j of each block is

$$\begin{aligned} A_{ij} = \sum_{pq} \bar{y}_{pq} y_{pq} \delta_p^i \delta_p^j &= \begin{cases} \sum\limits_{q\neq i} |y_{iq}^2|, & i=j \\ 0, & i\neq j \end{cases} \\ B_{ij} = \sum_{pq} \bar{y}_{pq} \bar{y}_{qp} \delta_p^i \delta_q^j &= \begin{cases} \bar{y}_{ij}\bar{y}_{ji}, & i\neq j \\ 0, & i=j \end{cases} \\ C_{ij} = \sum_{pq} y_{qp} y_{pq} \delta_q^i \delta_p^j &= \bar{B}_{ij} \\ D_{ij} = \sum_{pq} y_{pq} \bar{y}_{pq} \delta_q^i \delta_q^j &= A_{ij} \end{aligned} \tag{13}$$

We then write $\boldsymbol{J}^H\boldsymbol{J}$ in terms of the four $N_{\rm ant} \times N_{\rm ant}$ blocks as:

$$\boldsymbol{J}^H\boldsymbol{J} = \left[\begin{array}{c|c} \operatorname{diag} \sum\limits_{q\neq i} |y_{iq}^2| & \begin{cases} \bar{y}_{ij}\bar{y}_{ji}, & i\neq j \\ 0, & i=j \end{cases} \\ \hline \begin{cases} y_{ij}y_{ji}, & i\neq j \\ 0, & i=j \end{cases} & \operatorname{diag} \sum\limits_{q\neq i} |y_{iq}^2| \end{array}\right] \tag{14}$$

Equation A2 in the Appendix provides an example of $\boldsymbol{J}^H\boldsymbol{J}$ for the 3-antenna case.

The other component of the LM/GN equations (Eq. 4) is the $\boldsymbol{J}^H\breve{\boldsymbol{r}}$ term. This will be a column vector of length $2N_{\rm ant}$. We can write this as a stack of two $N_{\rm ant}$-vectors:

$$\boldsymbol{J}^H\breve{\boldsymbol{r}} = -\begin{bmatrix} \sum\limits_{pq} \bar{y}_{pq} r_{pq} \delta_p^i \\ \sum\limits_{pq} y_{qp} r_{pq} \delta_q^i \end{bmatrix} = -\begin{bmatrix} \sum\limits_{q\neq i} \bar{y}_{iq} r_{iq} \\ \sum\limits_{q\neq i} y_{iq} \bar{r}_{iq} \end{bmatrix} \begin{matrix} \}_{i=1\ldots N_{\rm ant}} \\ \}_{i=1\ldots N_{\rm ant}}, \end{matrix} \tag{15}$$

with the second equality established by swapping p and q in the bottom sum, and making use of $r_{pq} = \bar{r}_{qp}$. Clearly, the bottom half of the vector is the conjugate of the top:

$$\boldsymbol{J}^H\breve{\boldsymbol{r}} = -\begin{bmatrix} \boldsymbol{c} \\ \bar{\boldsymbol{c}} \end{bmatrix},\quad c_i = \sum_{q\neq i} \bar{y}_{iq} r_{iq}. \tag{16}$$

3.2 Computing the parameter update

By analogy with the augmented residuals vector $\breve{\boldsymbol{r}}$, we can express the data and model visibilities as $2N_{\rm bl}$-vectors, using the compound index $[pq]$ $(p \neq q)$:

$$\breve{\boldsymbol{d}} = [d_{pq}],\quad \breve{\boldsymbol{v}} = [g_p m_{pq} \bar{g}_q],\quad \breve{\boldsymbol{r}} = \breve{\boldsymbol{d}} - \breve{\boldsymbol{v}}$$

As noted by **?**, we have the wonderful property that

$$\breve{\boldsymbol{v}} = -\boldsymbol{J}\breve{\boldsymbol{g}},\quad \text{where } \breve{\boldsymbol{g}} = \begin{bmatrix} \boldsymbol{g} \\ \bar{\boldsymbol{g}} \end{bmatrix} \tag{17}$$

which basically comes about due to the RIME being bilinear with respect to $\boldsymbol{g}$ and $\bar{\boldsymbol{g}}$. Consider now the GN update step (Eq. 4 with $\lambda = 0$):

$$\breve{\boldsymbol{g}}_{k+1} = \breve{\boldsymbol{g}}_k + \delta\breve{\boldsymbol{g}} = \breve{\boldsymbol{g}}_k - (\boldsymbol{J}^H\boldsymbol{J})^{-1}\boldsymbol{J}^H(\breve{\boldsymbol{d}} - \breve{\boldsymbol{v}})$$

Substituting Eq. 17 into this, we arrive at

$$\breve{\boldsymbol{g}}_{k+1} = -(\boldsymbol{J}^H\boldsymbol{J})^{-1}\boldsymbol{J}^H\breve{\boldsymbol{d}}, \tag{18}$$

which indicates that the updated parameters at each iteration can be computed directly from the data, thus obviating the need for computing residuals. This can represent substantial algorithmic savings, and we will return to the idea again.

3.3 Time/frequency solution intervals

A common use case (especially in low-SNR scenarios) is to employ larger solution intervals. That is, we measure multiple visibilities per each baseline pq, across an interval of timeslots and frequency channels, then obtain complex gain solutions that are constant across each interval. The minimization problem of Eq. 6 can then be re-written as

$$\min_{\boldsymbol{g}} \sum_{pqs} |r_{pqs}|^2,\quad r_{pqs} = d_{pqs} - g_p m_{pqs} \bar{g}_q, \tag{19}$$

where $s = 1, ..., N_s$ is a sample index enumerating all the samples within the time/frequency solution interval. We can repeat the derivations above using $[pqs]$ as a single compound index. Instead of having shape $2N_{\rm bl} \times 2N_{\rm ant}$, the Jacobian will have a shape of $2N_{\rm bl}N_s \times 2N_{\rm ant}$, and the residual vector will have a length of $2N_{\rm bl}N_s$. In deriving the $\boldsymbol{J}^H\boldsymbol{J}$ term, the sums in Eq. 12 must be taken over all pqs rather

than just pq. Defining the usual shorthand of $y_{pqs} = m_{pqs}\bar{g}_q$, we then have:

$$\boldsymbol{J}^H\boldsymbol{J} = \left[\begin{array}{l|l} \operatorname{diag}\sum\limits_{q\neq i,s}|y_{iqs}^2| & \nearrow^H \\ \hline \begin{cases}\sum\limits_s y_{ijs}y_{jis}, & i\neq j \\ 0, & i=j\end{cases} & \searrow \end{array}\right], \tag{20}$$

where the symbols $\searrow$ and $\nearrow^H$ represent a copy and a copy-transpose of the appropriate matrix block (as per the structure of Eq. 12). Likewise, the $\boldsymbol{J}^H\breve{\boldsymbol{r}}$ term can be written as:

$$\boldsymbol{J}^H\breve{\boldsymbol{r}} = -\left[\begin{array}{c} \sum\limits_{q\neq i,s}\bar{y}_{iqs}r_{iqs} \\ \hline \downarrow^H \end{array}\right]. \tag{21}$$

3.4 Weighting

Although Laurent et al. (2012) do not mention this explicitly, it is straightforward to incorporate weights into the complex LS problem. Equation 1 is reformulated as

$$\min_{\boldsymbol{z}} ||\boldsymbol{W}\breve{\boldsymbol{r}}(\boldsymbol{z},\bar{\boldsymbol{z}})||_F, \tag{22}$$

where $\boldsymbol{W}$ is an $M\times M$ weights matrix (usually, the inverse of the data covariance matrix $\boldsymbol{C}$). This then propagates into the LM equations as

$$\delta\breve{\boldsymbol{z}} = -(\boldsymbol{J}^H\boldsymbol{W}\boldsymbol{J} + \lambda\mathbb{I})^{-1}\boldsymbol{J}^H\boldsymbol{W}\breve{\boldsymbol{r}}_k. \tag{23}$$

Adding weights to Eqs. 20 and 21, we arrive at the following:

$$\boldsymbol{J}^H\boldsymbol{W}\boldsymbol{J} = \left[\begin{array}{l|l} \operatorname{diag}\sum\limits_{q\neq i,s} w_{iqs}|y_{iqs}^2| & \nearrow^H \\ \hline \begin{cases}\sum\limits_s w_{ijs}y_{ijs}y_{jis}, & i\neq j \\ 0 \quad , & i=j\end{cases} & \searrow \end{array}\right] \tag{24}$$

$$\boldsymbol{J}^H\boldsymbol{W}\breve{\boldsymbol{r}} = -\left[\begin{array}{c} \sum\limits_{q\neq i,s} w_{iqs}\bar{y}_{iqs}r_{iqs} \\ \hline \downarrow^H \end{array}\right]. \tag{25}$$

3.5 Direction-dependent calibration

Let us apply the same formalism to the direction-dependent (DD) calibration problem. We reformulate the sky model as a sum of $N_{\rm dir}$ sky components, each with its own DD gain. It has been common practice to do DD gain solutions on larger time/frequency intervals than DI solutions, both for SNR reasons, and because short intervals lead to underconstrained solutions and suppression of unmodelled sources. We therefore incorporate solution intervals into the equations from the beginning. The minimization problem becomes:

$$\min_{\boldsymbol{g}}\sum_{pqs}|r_{pqs}|^2, \quad r_{pqs} = d_{pqs} - \sum_{d=1}^{N_{\rm dir}} g_p^{(d)}m_{pqs}^{(d)}\bar{g}_q^{(d)}. \tag{26}$$

It's obvious that the Jacobian corresponding to this this problem is very similar to the one in Eq. 11, but instead of having shape $2N_{\rm bl}\times 2N_{\rm ant}$, this will have a shape of $2N_{\rm bl}N_s\times 2N_{\rm ant}N_{\rm dir}$. We now treat $[pqs]$ and $[jd]$ as compound indices:

$$\boldsymbol{J} = -\left[\begin{array}{cc} \overbrace{m_{pqs}^{(d)}\bar{g}_q^{(d)}\delta_p^j}^{\substack{j=1\ldots N_{\rm ant}\\ d=1\ldots N_{\rm dir}}} & \overbrace{g_p^{(d)}m_{pqs}^{(d)}\delta_q^j}^{\substack{j=1\ldots N_{\rm ant}\\ d=1\ldots N_{\rm dir}}} \end{array}\right] \begin{array}{l} \}\ {\scriptstyle [pq]=1\ldots 2N_{\rm bl}\ (p\neq q)} \\ }\ {\scriptstyle s=1\ldots N_s} \end{array}$$

Every antenna j and direction d will correspond to a column in $\boldsymbol{J}$, but the specific order of the columns (corresponding to the order in which we place the $g_p^{(d)}$ elements in the augmented parameter vector $\breve{\boldsymbol{g}}$) is completely up to us.

Consider now the $\boldsymbol{J}^H\boldsymbol{J}$ product. This will consist of 2×2 blocks, each of shape $[N_{\rm ant}N_{\rm dir}]^2$. Let's use i,c to designate the rows within each block, j,d to designate the columns, and define $y_{pqs}^{(d)} = m_{pqs}^{(d)}\bar{g}_q^{(d)}$. The $\boldsymbol{J}^H\boldsymbol{J}$ matrix will then have the following block structure:

$$\boldsymbol{J}^H\boldsymbol{J} = \begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}^H \\ \boldsymbol{B} & \boldsymbol{A}\end{bmatrix} = \left[\begin{array}{l|l} \delta_j^i\sum\limits_{q\neq i,s}\bar{y}_{iqs}^{(c)}y_{iqs}^{(d)} & \nearrow^H \\ \hline \begin{cases}\sum\limits_s y_{jis}^{(c)}y_{ijs}^{(d)}, & i\neq j \\ 0 & i=j\end{cases} & \searrow \end{array}\right], \tag{27}$$

while the $\boldsymbol{J}^H\breve{\boldsymbol{r}}$ term will be a vector of length $2N_{\rm ant}N_{\rm dir}$, with the bottom half again being a conjugate of the top half. Within each half, we can write out the element corresponding to antenna j, direction d:

$$\boldsymbol{J}^H\breve{\boldsymbol{r}} = -\begin{bmatrix}\boldsymbol{c} \\ \bar{\boldsymbol{c}}\end{bmatrix}, \quad c_{jd} = \sum_{q\neq j,s}\bar{y}_{jqs}^{(d)}r_{jqs}. \tag{28}$$

Finally, let us note that the property of Eq. 17 also holds for the direction-dependent case. It is easy to see that

$$\breve{\boldsymbol{v}} = \left[\sum_{d=1}^{N_{\rm dir}} g_p^{(d)}m_{pqs}^{(d)}\bar{g}_q^{(d)}\right] = -\boldsymbol{J}\breve{\boldsymbol{g}}. \tag{29}$$

4 INVERTING $\boldsymbol{J}^H\boldsymbol{J}$ AND SEPARABILITY

In principle, implementing one of the flavours of calibration above is "just" a matter of plugging Eqs. 14+16, 20+21, 24+25 or 27+28 into one the algorithms defined in Appendix B. Note, however, that both the GN and LM algorithms hinge around inverting a large matrix. This will have a size of $2N_{\rm ant}$ or $2N_{\rm ant}N_{\rm dir}$ squared, for the DI or DD case respectively. With a naive implementation of matrix inversion, which scales cubically, algorithmic costs become dominated by the $O(N_{\rm ant}^3)$ or $O(N_{\rm ant}^3N_{\rm dir}^3)$ cost of inversion.

In this section we investigate approaches to simplifying the inversion problem by approximating $\boldsymbol{J}^H\boldsymbol{J}$ by some form of (block-)diagonal matrix. Such approximation is equivalent to separating the optimization problem into subsets of parameters that are treated as independent. We will show that some of these approximations are similar to or even fully equivalent to previously proposed algorithms, while others produce new algorithmic variations.

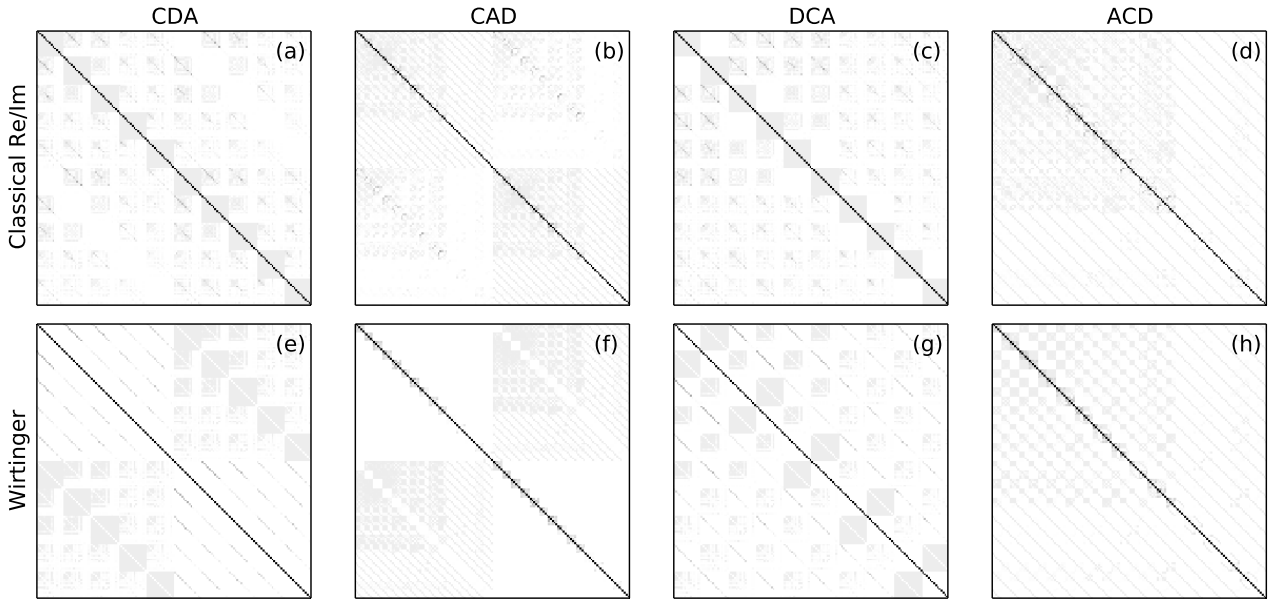

Figure 1. A graphical representation of $\boldsymbol{J}^H\boldsymbol{J}$ for a case of 40 antennas and 5 directions. Each pixel represents the amplitude of a single matrix element. (a) Conventional real-only Jacobian constructed by taking the partial derivatives w.r.t. the real and imaginary parts of the gains; (b) full complex Jacobian where the gains are ordered by direction major, antenna minor; (c) full complex Jacobian where the gains are ordered by antenna major, direction minor. Note that (c) can also be taken to represent the direction-independent case, with each 5×5 block becoming one pixel.

4.1 Diagonal approximation and Stefcal

Let us first consider the DI case. The structure of $\boldsymbol{J}^H\boldsymbol{J}$ in Eq. 14 suggests that it is diagonally dominant (especially for larger $N_{\rm ant}$), as each diagonal element is a coherent sum of $N_{\rm ant}$ amplitude-squared y-terms, while the off-diagonal elements are either zero or a product of two y terms. This is graphically illustrated in Fig. 1(c). It is therefore not unreasonable to approximate $\boldsymbol{J}^H\boldsymbol{J}$ with a diagonal matrix for purposes of inversion (or equivalently, making the assumption that the problem is separable per antenna). This makes the costs of matrix inversion negligible – $O(N_{\rm ant})$ operations, as compared to the $O(N_{\rm ant}^2)$ cost of computing the diagonal elements of the Jacobian in the first place. The price of using an approximate inverse for $\boldsymbol{J}^H\boldsymbol{J}$ is a less accurate update step, so we can expect to require more iterations before convergence is reached.

Combining this approximation with GN optimization, we may use Eq. 18 to write out the GN update step:

$$\breve{\boldsymbol{g}}_{k+1} = -(\boldsymbol{J}^H\boldsymbol{J})^{-1}\boldsymbol{J}^H\breve{\boldsymbol{d}},$$

Exploiting the redundancy of the equations, we only need to compute the update to $\boldsymbol{g}$ and not $\bar{\boldsymbol{g}}$. Since the diagonal approximation consists of assuming $\boldsymbol{B} = 0$ in Eq. 12, we can write the update to $\boldsymbol{g}$ as:

$$\boldsymbol{g}_{k+1} = -\mathrm{ul}(\boldsymbol{J}^H\boldsymbol{J})^{-1}\mathrm{top}(\boldsymbol{J}^H)\breve{\boldsymbol{d}},$$

where $\mathrm{ul}\,\boldsymbol{X}$ designates the upper-left quadrant of matrix $\boldsymbol{X}$, and $\mathrm{top}\,\boldsymbol{X}$ designates its top half. The per-element expression is then

$$g_{p,k+1} = \big(\sum_{q\neq p}\bar{y}_{pq}y_{pq}\big)^{-1}\sum_{q\neq p}\bar{y}_{pq}d_{pq}. \qquad (30)$$

Note that this expression does not contain the residual term, but only the data term. This eliminates the need to recompute the residuals at each step (except perhaps for purposes of checking convergence). Equation 30 is identical to the update step proposed by Salvini & Wijnholds (2014) for the Stefcal algorithm, and by Mitchell et al. (2008) for MWA calibration. Note that these authors arrive at the result from a different direction, by treating Eq. 7 as a function of $\boldsymbol{g}$ only, and completely ignoring the conjugate term. The resulting complex Jacobian (Eq. 10) then has null off-diagonal blocks, and $\boldsymbol{J}^H\boldsymbol{J}$ becomes diagonal.

Interestingly, if apply the diagonal $\boldsymbol{J}^H\boldsymbol{J}$ approximation to LM optimization (Eq. 5), we can derive the following update equation (in full analogy with Eq. 18):

$$\boldsymbol{g}_{k+1} = \frac{\lambda}{1+\lambda}\boldsymbol{g}_k - \frac{1}{1+\lambda}\mathrm{ul}(\boldsymbol{J}^H\boldsymbol{J})^{-1}\mathrm{top}(\boldsymbol{J}^H)\breve{\boldsymbol{d}}, \qquad (31)$$

which for $\lambda = 1$ essentially becomes the basic average-update step of Stefcal.

Establishing the equivalence between Stefcal and complex optimization with a diagonally-approximated $\boldsymbol{J}^H\boldsymbol{J}$ is very useful for our purposes, since the convergence properties of Stefcal have been thoroughly explored by Salvini & Wijnholds (2014), and we can therefore hope to apply these lessons here. In particular, these authors have shown that a direct application of Eq. 30 leads to very slow convergence, whereas averaging every pair of updates leads to faster convergence. They also propose a number of variations of the algorithm, all of which are directly applicable to the above.

Finally, let us note in passing that the update step of Eq. 30 is embarassingly parallel, in the sense that the update for each antenna is computed entirely independently.

4.2 Separability of the direction-dependent case

Now consider the problem of inverting $\boldsymbol{J}^H\boldsymbol{J}$ in the DD case. This is a massive matrix, and a brute force approach would scale as $O(N_{\rm dir}^3 N_{\rm ant}^3)$. We can, however, adopt a few approximations. Note again that we are free to reorder our augmented parameter vector (which contains both $\boldsymbol{g}$ and $\bar{\boldsymbol{g}}$), as long as we reorder the rows and columns of $\boldsymbol{J}^H\boldsymbol{J}$ accordingly.

Let us consider a number of orderings for $\breve{\boldsymbol{g}}$:

- conjugate major, direction, antenna minor (CDA):

$$[g_1^{(1)} \dots g_{N_{\rm ant}}^{(1)}, g_1^{(2)} \dots g_{N_{\rm ant}}^{(2)}, g_1^{(3)} \dots g_{N_{\rm ant}}^{(N_{\rm dir})}, \bar{g}_1^{(1)} \dots \bar{g}_{N_{\rm ant}}^{(1)} \dots]^T$$

- conjugate, antenna, direction (CAD):

$$[g_1^{(1)} \dots g_1^{(N_{\rm dir})}, g_2^{(1)} \dots g_2^{(N_{\rm dir})}, g_3^{(1)} \dots g_{N_{\rm ant}}^{(N_{\rm dir})}, \bar{g}_1^{(1)} \dots]^T$$

- direction, conjugate, antenna (DCA):

$$[g_1^{(1)} \dots g_{N_{\rm ant}}^{(1)}, \bar{g}_1^{(1)} \dots \bar{g}_{N_{\rm ant}}^{(1)}, g_1^{(2)} \dots g_{N_{\rm ant}}^{(2)}, \bar{g}_1^{(2)} \dots]^T$$

- antenna, conjugate, direction (ACD):

$$[g_1^{(1)} \dots g_1^{(N_{\rm dir})}, \bar{g}_1^{(1)} \dots \bar{g}_1^{(N_{\rm dir})}, g_2^{(1)} \dots g_2^{(N_{\rm dir})}, \bar{g}_2^{(1)} \dots]^T$$

Figure 1 graphically illustrates two of these orderings: (b) CDA and (c) CAD.

At this point we may derive a whole family of DD calibration algorithms – there are many ways to skin a cat. Each algorithm is defined by picking an ordering for $\breve{\boldsymbol{g}}$, then examining the corresponding $\boldsymbol{J}^H\boldsymbol{J}$ structure and specifying an approximate matrix inversion mechanism, then applying GN or LM optimization. Let us now work through a couple of examples.

4.2.1 DCA: separating by direction

Let us first consider the DCA ordering. The $\boldsymbol{J}^H\boldsymbol{J}$ term can be split into $N_{\rm dir} \times N_{\rm dir}$ blocks:

$$\boldsymbol{J}^H\boldsymbol{J} = \begin{bmatrix} \mathcal{J}_1^1 & \dots & \mathcal{J}_1^{N_{\rm dir}} \\ \vdots & & \vdots \\ \mathcal{J}_{N_{\rm dir}}^1 & \dots & \mathcal{J}_{N_{\rm dir}}^{N_{\rm dir}} \end{bmatrix}, \tag{32}$$

where the structure of each $2N_{\rm ant} \times 2N_{\rm ant}$ block at row c, column d, is exactly as given by Eq. 27.

The on-diagonal ("same-direction") blocks $\mathcal{J}_d^d$ will have the same structure as in the DI case (Eq. 14 or Eq. A2). Consider now the off-diagonal ("cross-direction") blocks $\mathcal{J}_c^d$. Their non-zero elements can take one of two forms:

$$\sum_{q\neq i,s} \bar{y}_{iqs}^{(c)} y_{iqs}^{(d)} = \sum_{q\neq i} g_q^{(c)} \bar{g}_q^{(d)} \sum_s \bar{m}_{iqs}^{(c)} m_{iqs}^{(d)}$$

or

$$\sum_s y_{jis}^{(c)} y_{ijs}^{(d)} = \bar{g}_i^{(c)} \bar{g}_j^{(d)} \sum_s \bar{m}_{ijs}^{(c)} m_{ijs}^{(d)} \tag{33}$$

A common element of both is essentially a dot product of sky model components. This is a measure of how "non-orthogonal" the components are:

$$X_{pq}^{(cd)} = \left\langle \boldsymbol{m}_{pq}^{(\rm c)}, \boldsymbol{m}_{pq}^{(\rm d)} \right\rangle = \sum_s m_{pqs}^{(c)} \bar{m}_{pqs}^{(d)}. \tag{34}$$

We should now note that each model component will typically correspond to a source of limited extent. This can be expressed as

$$m_{pqt\nu}^{(d)} = S_{pqt\nu}^{(d)} k_{pqt\nu}^{(d)},$$

where the term S represents the visibility of that sky model component if placed at phase centre (usually only weakly dependent on t, ν – in the case of a point source, for example, S is just a constant flux term), while the term

$$k_{pqt,\nu}^{(d)} = e^{-2\pi i (\boldsymbol{u}_{pq}(t)\cdot\boldsymbol{\sigma}_d)\nu/c}, \quad \boldsymbol{\sigma}_d = [l_d, m_d, n_d - 1]^T,$$

represents the phase rotation to direction $\boldsymbol{\sigma}_d$ (where lmn are the corresponding direction cosines), given a baseline vector as a function of time $\boldsymbol{u}_{pq}(t)$. We can then approximate the sky model dot product above as

$$X_{pq}^{(cd)} = S_{pq}^{(c)} S_{pq}^{(d)} \sum_s e^{-2\pi i [\boldsymbol{u}_{pq}(t)\cdot(\boldsymbol{\sigma}_c - \boldsymbol{\sigma}_d)]\nu/c}$$

The sum over samples s is essentially just an integral over a complex fringe. We may expect this to be small (i.e. the sky model components to be more orthogonal) if the directions are well-separated, and also if the sum is taken over longer time and frequency intervals.

If we now assume that the sky model components are orthogonal or near-orthogonal, then we may treat the "cross-direction" blocks of the $\boldsymbol{J}^H\boldsymbol{J}$ matrix in Eq. 32 as null. The problem is then separable by direction, and $\boldsymbol{J}^H\boldsymbol{J}$ becomes block-diagonal:

$$\boldsymbol{J}^H\boldsymbol{J} \approx \begin{bmatrix} \mathcal{J}_1^1 & & 0 \\ & \ddots & \\ 0 & & \mathcal{J}_{N_{\rm dir}}^{N_{\rm dir}} \end{bmatrix}, \tag{35}$$

The inversion complexity then reduces to $O(N_{\rm dir} N_{\rm ant}^3)$, which, for large numbers of directions, is a huge improvement on $O(N_{\rm dir}^3 N_{\rm ant}^3)$. Either GN and LM optimization may now be applied.

4.2.2 COHJONES: separating by antenna

A complementary approach is to separate the problem by antenna instead. Consider the CAD ordering (Fig. 1c). The top half of $\boldsymbol{J}^H\boldsymbol{J}$ then has the following block structure (and the bottom half is its symmetric conjugate):

$$\boldsymbol{J}^H\boldsymbol{J} = \begin{bmatrix} \boldsymbol{A}_1^1 & & 0 & \boldsymbol{B}_1^1 & \dots & \boldsymbol{B}_1^{N_{\rm ant}} \\ & \ddots & & \vdots & & \vdots \\ 0 & & \boldsymbol{A}_{N_{\rm ant}}^{N_{\rm ant}} & \boldsymbol{B}_{N_{\rm ant}}^1 & \dots & \boldsymbol{B}_{N_{\rm ant}}^{N_{\rm ant}} \end{bmatrix}, \quad (36)$$

that is, its left half is block-diagonal, consisting of $N_{\rm dir} \times N_{\rm dir}$ blocks (which follows from Eq. 27), while its right half consists of elements of the form given by Eq. 33.

By analogy with the Stefcal approach, we may assume $\boldsymbol{B}_j^i \approx 0$, i.e. treat the problem as separable by antenna. The $\boldsymbol{J}^H\boldsymbol{J}$ matrix then becomes block-diagonal, and we only need to compute the true matrix inverse of each $\boldsymbol{A}_i^i$. The inversion problem then reduces to $O(N_{\rm dir}^3 N_{\rm ant})$ in complexity, and either LM or GN optimization may be applied.

For GN, the update step may be computed in direct analogy to Eq. 30:

$$\boldsymbol{g}_{k+1} = -\boldsymbol{A}^{-1}\mathrm{top}(\boldsymbol{J}^H)\breve{\boldsymbol{d}}.$$

Note that for LM, the analogy is only approximate:

$$\breve{\boldsymbol{g}}_{k+1} \approx \frac{\lambda}{1+\lambda}\breve{\boldsymbol{g}}_k - \frac{1}{1+\lambda}\boldsymbol{A}^{-1}\mathrm{top}(\boldsymbol{J}^H)\breve{\boldsymbol{d}},$$

since $\boldsymbol{A}$ is only approximately diagonal.

Following Salvini & Wijnholds (2014), we can expect to improve convergence by averaging every pair of updates together. A variant of this approach, using GN optimization and a Stefcal-style solution averaging, has been implemented as the CohJones (complex half-Jacobian optimization for n-directional estimation) algorithm[2], the results of which applied to simulated data are presented below.

4.2.3 ALLJONES: Separating by everything

Perhaps the biggest simplification available is to start with CDA or CAD ordering, and assume a Stefcal-style diagonal approximation the entirety of $\boldsymbol{J}^H\boldsymbol{J}$. The matrix then becomes purely diagonal, matrix inversion reduces to $O(N_{\rm dir}N_{\rm ant})$ in complexity, and algorithmic cost becomes dominated by the $O(N_{\rm dir}N_{\rm ant}^2)$ process of computing the diagonal elements of $\boldsymbol{J}^H\boldsymbol{J}$. The GN update step can then computed in direct analogy to Eq. 30:

$$\boldsymbol{g}_{k+1} = -\mathrm{ul}(\boldsymbol{J}^H\boldsymbol{J})^{-1}\mathrm{top}(\boldsymbol{J}^H)\breve{\boldsymbol{d}},$$

and the per-element expression is

$$g_{p,k+1}^{(d)} = \Big(\sum_{q\neq p,s} \bar{y}_{pqs}^{(d)} y_{pqs}^{(d)}\Big)^{-1} \sum_{q\neq p,s} \bar{y}_{pqs}^{(d)} d_{pqs}. \quad (37)$$

4.3 Convergence and trade-offs

In the authors' experience (Smirnov 2013), an LM implementation of the DI calibration problem converges in fewer steps than Stefcal, but is almost always slower in terms of "wall time" due to the more expensive iterations. This is a typical "cheap–approximate" versus "expensive–accurate" trade-off, consistent with the expected scaling properties of these algorithms. The diagonal (separable by antenna) approximation makes for extremely cheap matrix inversions, but leads to less accurate steps.

With the DD flavours described here, we now have even more ways of making the trade-off:

- Brute-force LM or GN comes at a cost of $O(N_{\rm ant}^3 N_{\rm dir}^3)$ for the matrix inversion, but (in the authors' experience) requires very few (on the order of 10) iterations to converge,
- The AllJones algorithm is the cheapest, with $O(N_{\rm ant}^2 N_{\rm dir})$ performance (dominated by matrix construction rather than inversion), but offers the least accurate update step, since interactions between antennas and directions are completely ignored for the purposes of computing the update. This algorithm has not been tried in practice, and its convergence properties are unknown, but we can expect it to require very many iterations.
- The CohJones algorithm offers $O(N_{\rm ant}^2 N_{\rm dir} + N_{\rm ant} N_{\rm dir}^3)$ performance. It ignores interactions between antennas, but considers every direction at once. Practical experience has shown that the algorithm requires relatively few iterations to converge (see Sect. **??**).
- The DCA algorithm offers $O(N_{\rm ant}^3 N_{\rm dir})$ performance. It ignores interactions between directions, but considers every antenna at once. This algorithm has not been tried in practice, and its convergence properties are unknown.

Depending on $N_{\rm ant}$ and $N_{\rm dir}$, there may be regimes where one or the other algorithm has a computational advantage. This should be investigated in a future work.

4.4 Smoothing in time and frequency

From physical considerations, we know that gains do not vary arbitrarily as a functions of frequency and time. It can therefore be desirable to impose some sort of smoothness constraint on the solutions, which can improve conditioning, especially in low-SNR situations. A simple but crude way to do this is use solution intervals (Sect. 3.3), which gives a constant gain solution per interval, but produces non-physical jumps at the edge of each interval. Other approaches include aposteriori smoothing of solutions done on smaller intervals, as well as various filter-based algorithms (**?**).

Another way to impose smoothness combines the ideas of solution intervals (Eq. 19) and weighting (Eq. 22). At every time/frequency sample t_0, ν_0, we can postulate a weighted LS problem:

$$\min_{\boldsymbol{g}} \sum_{pqt\nu} w(t-t_0, \nu-\nu_0)|r_{pqt\nu}|^2, \quad (38)$$

where w is a smooth weighting kernel that upweighs samples at or near the current sample, and downweighs distant samples (e.g., a 2D Gaussian). The solutions for adjacent samples will be very close (since they are constrained by practically the same range of data points, with only a smooth change in weights), and the degree of smoothness can be controlled by tuning the width of the kernel.

On the face of it this approach is very expensive, since it entails an independent LS solution centred at every t_0, ν_0 sample. The diagonal approximation above, however, allows

[2] In fact it was the initial development of CohJones by **?** that directly led to the present work.

for a particularly elegant and efficient way of implementing this in practice. Consider the weighted equations of Eqs. 24 and 25, and replace the sample index s by t, ν. Under the diagonal approximation, each parameter update at t_0, ν_0 is computed as:

$$g_{p,k+1}(t_0,\nu_0) = \frac{\sum\limits_{q\neq p,t,\nu} w(t-t_0,\nu-\nu_0)\bar{y}_{pqt\nu}d_{pqt\nu}}{\sum\limits_{q\neq p,t,\nu} w(t-t_0,\nu-\nu_0)\bar{y}_{pqt\nu}y_{pqt\nu}}. \quad (39)$$

Looking at Eq. 39, it's clear that both sums represent a convolution. If we define two functions of t, ν:

$$\alpha_p(t,\nu) = \sum_{q\neq p} \bar{y}_{pqt\nu}d_{pqt\nu}, \quad \beta_p(t,\nu) = \sum_{q\neq p} \bar{y}_{pqt\nu}y_{pqt\nu},$$

then Eq. 39 corresponds to the ratio of two convolutions

$$g_{p,k+1}(t,\nu) = \frac{w \circ \alpha_p}{w \circ \beta_p}, \quad (40)$$

sampled over a discrete t, ν grid.

There is a very efficient way of implementing this in practice. Let's assume that d_{pq} and y_{pq} are loaded into memory and computed for a large chunk of t, ν values simultaneously (any practical implementation will probably need to do this anyway, if only to take advantage of vectorized math operations on modern CPUs and GPUs). The parameter update step is then also evaluated for a large chunk of t, ν, as are the α_p and β_p terms. We can then take advantage of highly optimized implementations of convolution (e.g. via FFTs) that are available on most computing architectures.

Iterating Eq. 40 to convergence at every t, ν slot, we obtain per-antenna arrays of gain solutions answrering Eq. 38. These solutions are smooth in frequency and time, with the degree of smoothness constrained by the kernel w.

5 THE FULLY POLARIZED CASE

To incorporate polarization, let us start by rewriting the basic RIME of Eq. 6 using 2×2 matrices (a full derivation may be found in Smirnov 2011):

$$\boldsymbol{D}_{pq} = \boldsymbol{G}_p \boldsymbol{M}_{pq} \boldsymbol{G}_q^H + \boldsymbol{N}_{pq}. \quad (41)$$

Here, $\boldsymbol{D}_{pq}$ is the *visibility matrix* observed by baseline pq, $\boldsymbol{M}_{pq}$ is the sky *coherency matrix*, $\boldsymbol{G}_p$ is the Jones matrix associated with antenna p, and $\boldsymbol{N}_{pq}$ is a noise matrix. Quite importantly, the visibility and coherency matrices are Hermitian: $\boldsymbol{D}_{pq} = \boldsymbol{D}_{qp}^H$, and $\boldsymbol{M}_{pq} = \boldsymbol{M}_{qp}^H$. The basic polarization calibration problem can be formulated as

$$\min_{\{\boldsymbol{G}_p\}} \sum_{pq} ||\boldsymbol{R}_{pq}||_F, \quad \boldsymbol{R}_{pq} = \boldsymbol{D}_{pq} - \boldsymbol{G}_p \boldsymbol{M}_{pq} \boldsymbol{G}_q^H \quad (42)$$

In principle, this can be recast into an ordinary complex NLLS problem (Eq. 1) by vectorizing each matrix in the above equation, and then deriving the complex Jacobian according to Eq. 3. We may, however, obtain a more elegant derivation by using a simple operator calculus, where the "atomic" elements are 2×2 matrices rather than scalars. We can then define the Jacobian more simply in terms of operators on such matrices.

5.1 Operator calculus

First, let us introduce the vectorization operator "vec" and its inverse in the usual way. For a 2×2 matrix:

$$\operatorname{vec} \boldsymbol{X} = \begin{bmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{bmatrix}, \quad \operatorname{vec}^{-1} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{bmatrix} = \boldsymbol{X},$$

This sets up a trivial isomorphism between the space of 2×2 complex matrices $\mathbb{C}^{2\times2}$ and the space $\mathbb{C}^4$. Any linear operator on $\boldsymbol{X} \in \mathbb{C}^{2\times2}$, which we'll write as $\mathcal{A}[\boldsymbol{X}]$, must then be equivalent to multiplication of $\operatorname{vec} \boldsymbol{X} \in \mathbb{C}^4$ by some 4×4 complex matrix $\boldsymbol{A}$. To put this formally, we can say that the vec operator induces an isomorphism $\mathcal{W}$ between $\mathbb{C}^{4\times4}$ and the space of linear operators on $\mathbb{C}^{2\times2}$:

$$\begin{aligned} &\mathcal{W} : \mathbb{C}^{4\times4} \leftrightarrow \operatorname{Lin}(\mathbb{C}^{2\times2}, \mathbb{C}^{2\times2}) \\ &\mathcal{W}(\boldsymbol{A})[\boldsymbol{X}] = \operatorname{vec}^{-1}(\boldsymbol{A} \operatorname{vec} \boldsymbol{X}) \\ &\mathcal{W}^{-1}(\mathcal{A})\boldsymbol{x} = \operatorname{vec}(\mathcal{A}[\operatorname{vec}^{-1} \boldsymbol{x}]) \end{aligned} \quad (43)$$

Of particular interest to us are two linear operators on $\mathbb{C}^{2\times2}$: right multiply by $\boldsymbol{A}$, and left multiply by $\boldsymbol{A}$, where $\boldsymbol{A}$ is a 2×2 complex matrix:

$$\begin{aligned} \mathcal{R}_{\boldsymbol{A}}[\boldsymbol{X}] &= \boldsymbol{X}\boldsymbol{A} \\ \mathcal{L}_{\boldsymbol{A}}[\boldsymbol{X}] &= \boldsymbol{A}\boldsymbol{X} \end{aligned}$$

The $\mathcal{W}$ isopmorphism ensures that these operators can be represented as multiplication of 4-vectors by a 4×4 matrix. Indeed, if $\boldsymbol{Y} = \mathcal{R}_{\boldsymbol{A}}[\boldsymbol{X}]$, then

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \end{bmatrix} = \operatorname{vec} \mathcal{R}_{\boldsymbol{A}}[\boldsymbol{X}] = \begin{bmatrix} a_{11} & a_{21} & 0 & 0 \\ a_{12} & a_{22} & 0 & 0 \\ 0 & 0 & a_{11} & a_{21} \\ 0 & 0 & a_{12} & a_{22} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{bmatrix} \quad (44)$$

Likewise,

$$\operatorname{vec} \mathcal{L}_{\boldsymbol{A}}[\boldsymbol{X}] = \begin{bmatrix} a_{11} & 0 & a_{12} & 0 \\ 0 & a_{11} & 0 & a_{12} \\ a_{21} & 0 & a_{22} & 0 \\ 0 & a_{21} & 0 & a_{22} \end{bmatrix} \operatorname{vec} \boldsymbol{X}. \quad (45)$$

We may repurpose the symbols $\mathcal{R}_{\boldsymbol{A}}$ and $\mathcal{L}_{\boldsymbol{A}}$ to also represent the 4×4 matrices themselves. It is trivial to see that

$$\mathcal{R}_{\boldsymbol{A}}^H = \mathcal{R}_{\boldsymbol{A}^H}, \quad \mathcal{L}_{\boldsymbol{A}}^H = \mathcal{L}_{\boldsymbol{A}^H}. \quad (46)$$

Other obvious properties are:

$$\mathcal{L}_{\boldsymbol{A}}\mathcal{L}_{\boldsymbol{B}} = \mathcal{L}_{\boldsymbol{AB}}, \quad \mathcal{R}_{\boldsymbol{A}}\mathcal{R}_{\boldsymbol{B}} = \mathcal{R}_{\boldsymbol{BA}}, \quad [\mathcal{R}_{\boldsymbol{A}}]^{-1} = \mathcal{R}_{\boldsymbol{A}^{-1}} \quad (47)$$

Note that Eq. 47 is equally valid whether interpreted in terms of chaining operators, or multipying the equivalent 4×4 matrices.

Finally, note that the transpose and Hermitian transpose operators in $\mathbb{C}^{2\times2}$ space correspond to a swapping of the second and third rows in $\mathbb{C}^4$ space:

$$\operatorname{vec} \boldsymbol{X}^T = \begin{bmatrix} x_{11} \\ x_{21} \\ x_{12} \\ x_{22} \end{bmatrix}, \quad \operatorname{vec} \boldsymbol{X}^H = \begin{bmatrix} \bar{x}_{11} \\ \bar{x}_{21} \\ \bar{x}_{12} \\ \bar{x}_{22} \end{bmatrix}. \tag{48}$$

5.2 Derivative operators and Jacobians

We can use the relations of Eq. 43 to induce a natural mapping between any matrix-valued function of a matrix argument $\boldsymbol{F}(\boldsymbol{G})$, $\boldsymbol{F} : \mathbb{C}^{2\times 2} \to \mathbb{C}^{2\times 2}$ and a corresponding vector-valued function of a vector, $\boldsymbol{f} : \mathbb{C}^4 \to \mathbb{C}^4$:

$$\boldsymbol{f}(\boldsymbol{g}) = \operatorname{vec} \boldsymbol{F}(\operatorname{vec}^{-1} \boldsymbol{g}).$$

Consider now the partial (and conjugate partial) Jacobians of $\boldsymbol{f}$ with respect to $\boldsymbol{g}$ (and $\bar{\boldsymbol{g}}$). These are 4×4 matrices (Eq. 8):

$$\boldsymbol{J}_g = [\partial f_i / \partial g_j], \quad \boldsymbol{J}_{\bar{g}} = [\partial f_i / \partial \bar{g}_j]$$

We can now use the isomorphism of Eq. 43 to formally define the corresponding "Wirtinger derivative operators" of $\boldsymbol{F}$ with respect to $\boldsymbol{G}$ and $\boldsymbol{G}^{\boldsymbol{H}}$ as

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{G}} = \mathcal{W}(\boldsymbol{J}_g), \quad \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{G}^H} = \mathcal{W}(\boldsymbol{J}_{\bar{g}}^T) \tag{49}$$

This formal definition will allow to write the Jacobians below in terms of matrices composed of operators. In particular, it is easy to see (just from considering the corresponding 4×4 matrices) that

$$\frac{\partial(\boldsymbol{GA})}{\partial \boldsymbol{G}} = \mathcal{R}_A, \quad \frac{\partial(\boldsymbol{AG}^{\boldsymbol{H}})}{\partial \boldsymbol{G}^{\boldsymbol{H}}} = \mathcal{L}_A.$$

5.3 Complex Jacobians for the polarized RIME

Let us apply the operator calculus developed above to Eq. 42. We have:

$$\frac{\partial \boldsymbol{R}_{pq}}{\partial \boldsymbol{G}_p} = -\mathcal{R}_{\boldsymbol{M}_{pq}\boldsymbol{G}_q^H}, \quad \text{and} \quad \frac{\partial \boldsymbol{R}_{pq}}{\partial \boldsymbol{G}_q^H} = -\mathcal{L}_{\boldsymbol{G}_p\boldsymbol{M}_{pq}}. \tag{50}$$

If we now stack all the gain matrices into one augmented "vector of matrices":

$$\breve{\boldsymbol{G}} = [\boldsymbol{G}_1, \ldots, \boldsymbol{G}_{N_{\text{ant}}}, \boldsymbol{G}_1^H, \ldots, \boldsymbol{G}_{N_{\text{ant}}}^H]^T, \tag{51}$$

then we may construct the top half of the full complex Jacobian operator in full analogy with the derivation of Eq. 10. We'll use the same "compound index" convention for pq. That is, $[pq]$ will represent a single index running through M values (i.e. enumerating all combinations of $p < q$).

$$\boldsymbol{J}_{\text{top}} = -\Big[\overbrace{\mathcal{R}_{\boldsymbol{M}_{pq}\boldsymbol{G}_q^H}\delta_p^j}^{j=1\ldots N_{\text{ant}}} \quad \overbrace{\mathcal{L}_{\boldsymbol{G}_p\boldsymbol{M}_{pq}}\delta_q^j}^{j=1\ldots N_{\text{ant}}} \Big] \ \}_{[pq]=1\ldots N_{\text{bl}}\ (p<q)} \tag{52}$$

Note that there are two fully equivalent ways to read the above equation. In operator notation, it specifies a linear operator $\boldsymbol{J}_{\text{top}}$ that maps a $2N_{\text{ant}}$-vector of 2×2 matrices to an N_{bl}-vector of 2×2 matrices. In conventional matrix notation, $\boldsymbol{J}_{\text{top}}$ is just a $4N_{\text{bl}} \times 8N_{\text{ant}}$ matrix; the above equation then specifies the structure of this matrix in terms of 4×4 blocks, where each block is the matrix equivalent of the appropriate $\mathcal{R}$ or $\mathcal{L}$ operator.

Consider now the bottom half of the Jacobian. In Eq. 3, this corresponds to the derivatives of the conjugate residual vector $\bar{\boldsymbol{r}}_k$, and can be constructed by conjugating and mirroring $\boldsymbol{J}_{\text{top}}$. Let us modify this construction by taking the derivative of the Hermitian transpose of the residuals. We construct the augmented residual vector of matrices as:

$$\breve{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{R}_{pq} \\ \boldsymbol{R}_{pq}^H \end{bmatrix} \begin{matrix} \}_{[pq]=1\ldots N_{\text{bl}}\ (p<q)} \\ \}_{[pq]=1\ldots N_{\text{bl}}\ (p<q)} \end{matrix} \tag{53}$$

Now, since $\boldsymbol{R}_{pq}^H = \boldsymbol{G}_q \boldsymbol{M}_{pq}^H \boldsymbol{G}_p^H$, we have

$$\frac{\partial \boldsymbol{R}_{pq}^H}{\partial \boldsymbol{G}_q} = -\mathcal{R}_{\boldsymbol{M}_{pq}^H\boldsymbol{G}_p^H}, \quad \text{and} \quad \frac{\partial \boldsymbol{R}_{pq}^H}{\partial \boldsymbol{G}_p^H} = -\mathcal{L}_{\boldsymbol{G}_q\boldsymbol{M}_{pq}^H}, \tag{54}$$

and we may write out the complete complex Jacobian as

$$\boldsymbol{J} = -\begin{bmatrix} \mathcal{R}_{\boldsymbol{M}_{pq}\boldsymbol{G}_q^H}\delta_p^j & \mathcal{L}_{\boldsymbol{G}_p\boldsymbol{M}_{pq}}\delta_q^j \\ \mathcal{R}_{\boldsymbol{M}_{pq}^H\boldsymbol{G}_p^H}\delta_q^j & \mathcal{L}_{\boldsymbol{G}_q\boldsymbol{M}_{pq}^H}\delta_p^j \end{bmatrix} \begin{matrix} \}_{[pq]=1\ldots N_{\text{bl}}\ (p<q)} \\ \}_{[pq]=1\ldots N_{\text{bl}}\ (p<q)} \end{matrix}$$

We may now make exactly the same observation as we did to derive Eq. 11, and rewrite both $\boldsymbol{J}$ and $\breve{\boldsymbol{R}}$ in terms of a single row block. The pq index will now run through $2N_{\text{bl}}$ values (i.e. enumerating all combinations of $p \neq q$):

$$\boldsymbol{J} = -\begin{bmatrix} \mathcal{R}_{\boldsymbol{M}_{pq}\boldsymbol{G}_q^H}\delta_p^j & \mathcal{L}_{\boldsymbol{G}_p\boldsymbol{M}_{pq}}\delta_q^j \end{bmatrix} \ \}_{[pq]=1\ldots 2N_{\text{bl}}\ (p\neq q)} \tag{55}$$

and

$$\breve{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{R}_{pq} \end{bmatrix} \ \}_{[pq]=1\ldots 2N_{\text{bl}}\ (p\neq q)} \tag{56}$$

This is in complete analogy to the derivations of the unpolarized case. For compactness, let us now define

$$\boldsymbol{Y}_{pq} = \boldsymbol{M}_{pq}\boldsymbol{G}_q^H,$$

and, noting that

$$\boldsymbol{Y}_{qp}^H = \boldsymbol{G}_p\boldsymbol{M}_{pq},$$

and using Eq. 46, we can now write out the $\boldsymbol{J}^H\boldsymbol{J}$ term, still expressed in terms of operators, as:

$$\boldsymbol{J}^H\boldsymbol{J} = \left[\begin{array}{c|c} \operatorname{diag} \sum_{q\neq i} \mathcal{R}_{\boldsymbol{Y}_{iq}\boldsymbol{Y}_{iq}^H} & \nearrow^H \\ \hline \left\{\begin{array}{cc} \mathcal{L}_{Y_{ji}}\mathcal{R}_{\boldsymbol{Y}_{ij}}, & i\neq j \\ 0, & i=j \end{array}\right. & \searrow \end{array}\right] \tag{57}$$

(Note that this makes use of the property $\mathcal{R}_{\boldsymbol{A}}\mathcal{R}_{\boldsymbol{B}} = \mathcal{R}_{\boldsymbol{BA}}$ and $\mathcal{L}_{\boldsymbol{A}}\mathcal{L}_{\boldsymbol{B}} = \mathcal{L}_{\boldsymbol{AB}}$.) Compare this result to Eq. 14.

For the $\boldsymbol{J}^H\breve{\boldsymbol{R}}$ term, we can directly apply the linear operators appearing in $\boldsymbol{J}^H$ to the matrices in $\breve{\boldsymbol{r}}$:

$$\boldsymbol{J}^H \breve{\boldsymbol{R}} = -\left[\begin{array}{c} \sum_{pq} \boldsymbol{Y}^H_{pq} \boldsymbol{R}_{pq} \delta^i_p \\ \hline \sum_{pq} \boldsymbol{R}_{pq} \boldsymbol{Y}_{qp} \delta^i_q \end{array}\right] = -\left[\begin{array}{c} \sum_{q\neq i} \boldsymbol{Y}^H_{iq} \boldsymbol{R}_{iq} \\ \hline \downarrow^H \end{array}\right], \tag{58}$$

where the second equality is established by swapping the p and q indices. Unsurprisingly, and by analogy with Eq. 16, the bottom half of the vector is Hermitian with respect to the top.

5.4 Parameter updates and the diagonal approximation

The relation of Eq. 17 also apply in the fully-polarized case. It is easy to see that if we define the augmented data and model vectors of matrices as

$$\breve{\boldsymbol{D}} = [\boldsymbol{D}_{pq}], \quad \breve{\boldsymbol{V}} = [\boldsymbol{G}_p \boldsymbol{M}_{pq} \boldsymbol{G}^H_q],$$

then $\breve{\boldsymbol{V}} = -\boldsymbol{J}\breve{\boldsymbol{G}}$ holds, and the GN update step can be written as

$$\breve{\boldsymbol{G}}_{k+1} = -(\boldsymbol{J}^H \boldsymbol{J})^{-1} \boldsymbol{J}^H \breve{\boldsymbol{D}}.$$

To actually implement GN or LM optimization, we still need to invert the $\boldsymbol{J}^H\boldsymbol{J}$ matrix that is represented in operator form by Eq. 57. We have two options here. The brute-force numerical approach is to substitute the 4×4 matrix forms of $\mathcal{R}$ and $\mathcal{L}$ into the equation, thus resulting in conventional matrix, and then do a straightfoward matrix inversion. The second option is to use an analogue of the diagonal approximation described in Sect. 4.1. If we neglect the off-diagonal operators of Eq. 57, the operator form of the matrix is diagonal, and the problem becomes separable per antenna. The inverse operator corresponding to $\mathcal{R}_{\boldsymbol{A}}$ is, trivially, $\mathcal{R}_{\boldsymbol{A}^{-1}}$. We then arrive a simple per-antenna equation for the GN update step:

$$\boldsymbol{G}_{p,k+1} = \left[\sum_{q\neq p} \boldsymbol{Y}^H_{pq} \boldsymbol{D}_{pq}\right] \left[\sum_{q\neq p} \boldsymbol{Y}_{pq} \boldsymbol{Y}^H_{pq}\right]^{-1} \tag{59}$$

This is, once again, equivalent to the polarized StefCal update step proposed by Salvini & Wijnholds (2014).

5.5 Polarized direction-dependent calibration

Let us now briefly address the fully-polarized DD case. This can be done by direct analogy with Sect. 3.5, using the operator calculus developed above. As a result, we arrive at the following expression for $\boldsymbol{J}^H\boldsymbol{J}$:

$$\boldsymbol{J}^H \boldsymbol{J} = \left[\begin{array}{c} \delta^i_j \sum_{q\neq i,s} \mathcal{R}_{\boldsymbol{Y}^{(d)}_{iqs} \boldsymbol{Y}^{(c)H}_{iqs}} \quad \big| \nearrow^H \\ \hline \left\{\begin{array}{cc} \sum_s \mathcal{L}_{\boldsymbol{Y}^{(c)}_{jis}} \mathcal{R}_{\boldsymbol{Y}^{(d)}_{ijs}}, & i\neq j \\ 0 & i=j \end{array}\right| \searrow \end{array}\right], \tag{60}$$

using the normal shorthand of $\boldsymbol{Y}^{(d)}_{pqs} = \boldsymbol{M}^{(d)}_{pqs} \boldsymbol{G}^{(d)H}_q$. The $\boldsymbol{J}^H\breve{\boldsymbol{R}}$ term is then

$$\boldsymbol{J}^H \breve{\boldsymbol{R}} = -\left[\begin{array}{c} \sum_{q\neq i,s} \boldsymbol{Y}^{(d)H}_{iqs} \boldsymbol{R}_{iqs} \\ \hline \downarrow^H \end{array}\right]. \tag{61}$$

All the separability considerations of Sect. 4 now apply, and polarized versions of the algorithms referenced therein may be reformulated for the fully polarized case. For example

- If we assume separability by both direction and antenna, as in the AllJones algorithm, then the $\boldsymbol{J}^H\boldsymbol{J}$ matrix is fully diagonal in operator form, and the GN update step can be directly computed as

$$\boldsymbol{G}^{(d)}_{p,k+1} = \left[\sum_{q\neq p,s} \boldsymbol{Y}^{(d)H}_{pqs} \boldsymbol{D}_{pqs}\right] \left[\sum_{q\neq p,s} \boldsymbol{Y}^{(d)}_{pqs} \boldsymbol{Y}^{(d)H}_{pqs}\right]^{-1} \tag{62}$$

- If we only assume separability by antenna, as in the CohJones algorithm, then the $\boldsymbol{J}^H\boldsymbol{J}$ matrix becomes $4N_{\rm dir} \times 4N_{\rm dir}$-block-diagonal.

It is also straightfoward to add weights and/or sliding window averging to this formulation, as per Sect. 3.4 and 4.4.

Equations 60–61 can be considered the principal result of this work. They provide the necessary ingredients for implementing GN or LM methods for DD calibration, treating it as a fully complex optimization problem. The equations may be combined and approximated in different ways to produce different types of calibration algorithms.

Another interesting note is that Eq. 62 is embarassingly parallel, since the update step is completely separated by direction and antenna. This makes it particularly well-suited to implementation on massively parallel architectures such as GPUs.

6 OTHER DD ALGORITHMIC VARIATIONS

The mathematical framework developed above (in particular, Eqs. 60–61) provides a general description of the polarized DD calibration problem. Practical implementations of this hinge around inversion of a very large $\boldsymbol{J}^H\boldsymbol{J}$ matrix. The family of algorithms proposed in Sect. 4 takes different approaches to approximating this inversion. Their convergence properties are not yet well-understood; however we may note that the StefCal algorithm naturally emerges from this formulation as a specific case, and its convergence has been established by Salvini & Wijnholds (2014). This is encouraging, but ought not be treated as anything more than a strong pointer for the DD case. It is therefore well worth exploring other approximations to the problem. In this section we map out a few such options.

6.1 Feed forward

Salvini & Wijnholds (2014) propose variants of the Stefcal algorithm ("2-basic" and "2-relax") where the results of the update step (Eq. 59, in essense) are computed sequentially per antenna, with updated values for $\boldsymbol{G}_1, \ldots, \boldsymbol{G}_{k-1}$ fed forward into the equations for $\boldsymbol{G}_k$. This is shown to

substantially improve convergence, at the cost of sacrificing the embarassing paralellism by antenna. This technique is directly applicable to both the AllJones and CohJones algorithms.

The CohJones algorithm considers all directions simultaneously, but could still implement feed-forward by antenna. The AllJones algorithm (Eq. 62) could implement feed-forward by both antenna and direction. The optimal order for this, as well as whether in practice this actually improves convergence to justify the extra complexity, is an open issue that remains to be investigated.

6.2 Triangular approximation

The main idea of feed-forward is to take into account solutions for antennas (and/or directions) $1, ..., k-1$ when computing the solution for k. A related approach is to approximate the $\boldsymbol{J}^H\boldsymbol{J}$ matrix as block-triangular:

$$\boldsymbol{J}^H\boldsymbol{J} \approx \begin{bmatrix} \mathcal{J}_1^1 & 0 & \cdots & 0 \\ \mathcal{J}_2^1 & \mathcal{J}_2^2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ \mathcal{J}_N^1 & \mathcal{J}_N^2 & \cdots & \mathcal{J}_N^N \end{bmatrix}, \tag{63}$$

The inverse of this is also block triangular:

$$(\boldsymbol{J}^H\boldsymbol{J})^{-1} \approx \begin{bmatrix} \mathcal{K}_1^1 & 0 & \cdots & 0 \\ \mathcal{K}_2^1 & \mathcal{K}_2^2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ \mathcal{K}_N^1 & \mathcal{K}_N^2 & \cdots & \mathcal{K}_N^N \end{bmatrix}, \tag{64}$$

which can be computed using Gaussian elimination:

$$\begin{aligned} \mathcal{K}_1^1 &= [\mathcal{J}_1^1]^{-1} \\ \mathcal{K}_2^2 &= [\mathcal{J}_2^2]^{-1} \\ \mathcal{K}_2^1 &= -\mathcal{K}_2^2\mathcal{J}_2^1\mathcal{K}_1^1 \\ \mathcal{K}_3^3 &= [\mathcal{J}_3^3]^{-1} \\ \mathcal{K}_3^2 &= -\mathcal{K}_3^3\mathcal{J}_3^2\mathcal{K}_2^2 \\ \mathcal{K}_3^1 &= -\mathcal{K}_3^3[\mathcal{J}_3^1\mathcal{K}_1^1 + \mathcal{J}_3^2\mathcal{K}_2^1] \\ &\cdots \end{aligned}$$

From this, the GN or LM update steps may be derived directly.

6.3 Peeling

The *peeling* procedure was originally suggested by Noordam (2004) as a "kludge", i.e. an implementation of DD calibration using the DI functionality of existing packages. In a nutshell, this procedure solves for DD gains towards one source at a time, from brighter to fainter, by

(i) Rephasing the visibilities to place the source at phase centre;
(ii) Averaging over some time/frequency interval (to suppress the contribution of other sources);
(iii) Doing a standard solution for DI gains (which approximates the DD gains towards the source);
(iv) Subtracting the source from the visibilities using the obtained solutions;
(v) Repeating the procedure for the next source.

[h!]

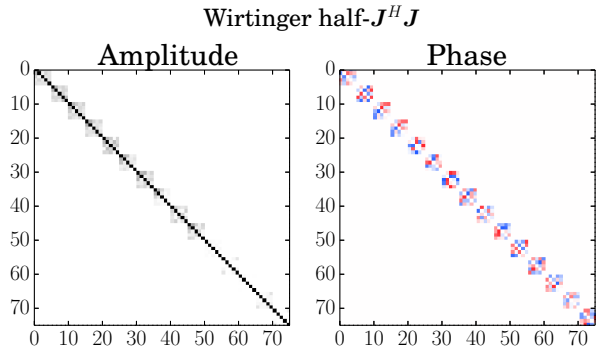

Figure 3. This figure shows amplitude (left panel) and phase (right panel) of the block-diagonal matrix $\boldsymbol{J}^H\boldsymbol{J}$ for the dataset described in the text. Each block corresponds to the different directions for each specific antenna. Its block structure make it easily invertible.

The term "peeling" comes from step (iv), since sources are "peeled" away one at a time[3].

Within the framework above, peeling can be considered as the ultimate feed forward approach. Peeling is essentially feed-forward by direction, except rather than taking one step over each direction in turn, each direction is iterated to full convergence before moving on to the next direction. The procedure can then be repeated beginning with the brightest source again, a second cycle tends to improve the solutions.

6.4 Exact matrix inversion

Better approximations to $(\boldsymbol{J}^H\boldsymbol{J})^{-1}$ (or a faster exact inverse) may exist. Consider again Fig. 1: in most cases, the matrix consists of four blocks, with the diagonal blocks being trivially invertable, and the off-diagonal blocks having a very specific structure. All the approaches discussed in this paper approximate the off-diagonal blocks by zero, and thus yield algorithms which converge to the solution via many cheap approximative steps. If a fast way to invert matrices of the off-diagonal type (faster than $O(N^3)$, that is) could be found, this could yield calibration algorithms that converge in fewer more accurate iterations.

7 TESTS ON SIMULATED DATA

In this section, CohJones is tested on simulated data for scalar Jones matrices only. In Sec. 7.1, the Jones matrices are constant in time. In that case only the convergence of CohJones can be studied. In Sec. 7.2, the Jones matrices vary in time.

[3] The term "peeling" has occasionally been misappropriated to describe other schemes, e.g. simultaneous independent DD gain solutions. We consider this a misuse: both the original formulation by Noordam (2004), and the word "peeling" itself, strongly implies dealing with one direction at a time.

[t!]

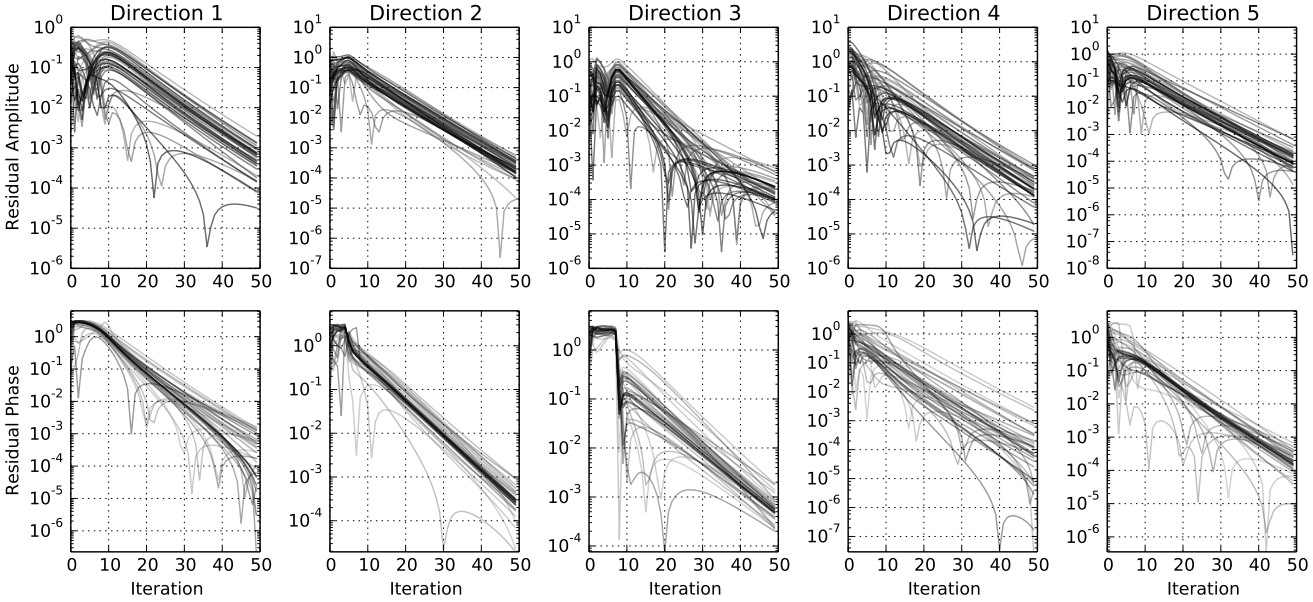

Figure 2. This plot shows the amplitude (top panels) and phase (bottom panels) of the difference between the estimated gains and the true (random) gains in the different directions for the different antenna (shaded greys full lines).

[t!]

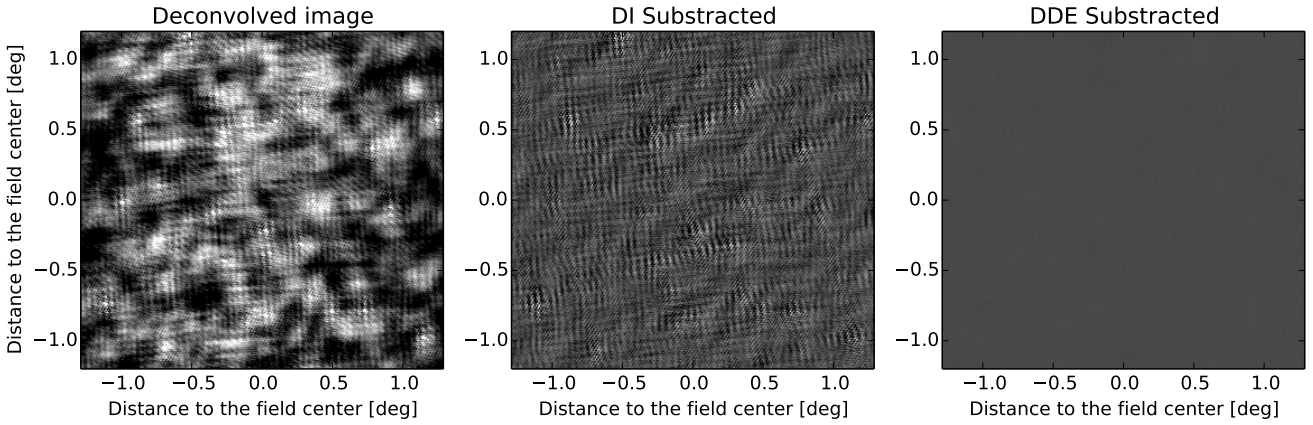

Figure 4. This figure shows the comparison between the deconvoled image (left), the residuals data after simple skymodel subtractions (center), and the residuals data after subtracting the sky model corrupted by the direction-dependent COHJONES estimated solution (right). The color scale is the same in all panels. In this simulation, COHJONES reduces the residual data level by a factor of ~ 30.

7.1 Time-constant Jones matrices

For this test, a visibility dataset is simulated assuming the Low Frequency Array (LOFAR) antenna layout. The phase center is located at $(\alpha, \delta) = (14^h11^m20.5^s, +52^o12'10.0'')$, the observing frequency is set 50 MHz, and time bins are 10 sec wide. To generate the visibilities, we use a sky model containing five sources, distributed in a cross, and separated by a degree. The gains applied to the antenna p in direction d are constant through time, and are taken at random along a normal distribution $g_p \sim \mathcal{N}(0,1) + i\mathcal{N}(0,1)$. The data vector is then built from all baselines, and a 20 minutes time chunk.

The corresponding matrix $\boldsymbol{J}^H\boldsymbol{J}$ is shown in Fig. 3. It is block diagonal, each block having size $n_d \times n_d$. The calibration solution convergence are shown in Fig. 2. It is important to note that the problems becomes better conditioned as the blocks of $\boldsymbol{J}^H\boldsymbol{J}$ become more diagonal. In that case COHJONES converges faster, and this happens (i) when more data are taken into account in the construction of the data vector or (ii) if the directions are put further away from each other.

[h!]

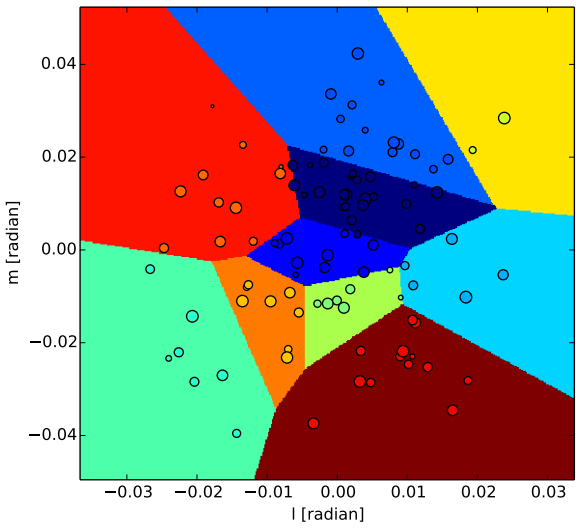

Figure 5. In order to conduct direction-dependent calibration, the sources of the sky model are clustered using a Voronoi tessellation algorithm.

7.2 Variable gains

In order to simulate a more realistic dataset, we use a 100 sources sky model which flux density is randomly distributed (uniform distribution). Noise is added to the visibilities at the 1% level of the total flux. The scalar Jones matrices are simulated assuming an ionospheric model consisting of a purely scalar, direction-dependent phase (an infinitesimally thin layer at a height of 100 km). The total electron content (TEC) values at a set of sample points are generated using Karhunen-Loeve decomposition (the spatial correlation is given by Kolmogorov turbulence, see van der Tol 2009). The sources are clustered in 10 directions using Voronoi tesselation (Fig. 5).

Fig. 4 shows the residuals data computed by subtracting the model data in the visibility domain, and the model data affected by DDEs. The residual data standard deviation reduces by a factor ~ 30 after COHJONES has been applied.

CONCLUSIONS

Other ways of inverting the matrix (someone clever may find one?)

Complexity O(chicken). Chicken!

Embarassingly parallel. GPU. Chicken!

Many variations possible, describes previous algorithms as special cases (approximations). Chicken!

Unifying framework? Chicken!

REFERENCES

Laurent S., van Barel M., de Lathauwer L., 2012, SIAM J. Optim., 22, 879

Mitchell D. A., Greenhill L. J., Wayth R. B., Sault R. J., Lonsdale C. J., Cappallo R. J., Morales M. F., Ord S. M., 2008, IEEE Journal of Selected Topics in Signal Processing, 2, 707

Noordam J. E., 2004, in Oschmann J. J. M., ed., Ground-based Telescopes Vol. 5489 of Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, LOFAR calibration challenges. p. 817

Salvini S., Wijnholds S., 2014, URSI GASS

Smirnov O. M., 2011, A&A, 527, A106

Smirnov O. M., 2013, StefCal: The fastest selfcal in the West, presentation at 3GC3 workshop (Port Alfred, February 2013), https://sites.google.com/a/ska.ac.za/3gc3/programmes/talk-slides

van der Tol S., 2009, PhD thesis, TU Delft

APPENDIX A: J AND J^HJ FOR THE THREE-ANTENNA CASE

To give a specific example of complex Jacobians, consider the 3 antenna case. Using the numbering convention for pq of 12, 13, 32, we obtain the following partial Jacobians (Eqs. 8 and 9):

$$\boldsymbol{J}_k = -\begin{bmatrix} m_{12}\bar{g}_2 & 0 & 0 \\ m_{13}\bar{g}_3 & 0 & 0 \\ 0 & m_{23}\bar{g}_3 & 0 \end{bmatrix}, \boldsymbol{J}_{\bar{k}} = -\begin{bmatrix} 0 & g_1 m_{12} & 0 \\ 0 & 0 & g_1 m_{13} \\ 0 & 0 & g_2 m_{23} \end{bmatrix}$$

We then get the following expression for the full complex Jacobian $\boldsymbol{J}$ (Eq. 11):

$$-\begin{bmatrix} m_{12}\bar{g}_2 & 0 & 0 & 0 & g_1 m_{12} & 0 \\ m_{13}\bar{g}_3 & 0 & 0 & 0 & 0 & g_1 m_{13} \\ 0 & m_{21}\bar{g}_1 & 0 & g_2 m_{21} & 0 & 0 \\ 0 & m_{23}\bar{g}_3 & 0 & 0 & 0 & g_2 m_{23} \\ 0 & 0 & m_{31}\bar{g}_1 & g_1 m_{31} & 0 & 0 \\ 0 & 0 & m_{32}\bar{g}_2 & 0 & g_3 m_{32} & 0 \end{bmatrix} \tag{A1}$$

Then, with the usual shorthand of $y_{pq} = m_{pq}\bar{g}_q$, the $\boldsymbol{J}^H\boldsymbol{J}$ term becomes:

$$\begin{bmatrix} y_{12}^2+y_{13}^2 & 0 & 0 & 0 & \bar{y}_{12}\bar{y}_{21} & \bar{y}_{13}\bar{y}_{31} \\ 0 & y_{12}^2+y_{23}^2 & 0 & \bar{y}_{12}\bar{y}_{21} & 0 & \bar{y}_{23}\bar{y}_{32} \\ 0 & 0 & y_{13}^2+y_{23}^2 & \bar{y}_{13}\bar{y}_{31} & \bar{y}_{23}\bar{y}_{32} & 0 \\ 0 & y_{12}y_{21} & y_{13}y_{31} & y_{12}^2+y_{13}^2 & 0 & 0 \\ y_{12}y_{21} & 0 & y_{23}y_{32} & 0 & y_{12}^2+y_{23}^2 & 0 \\ y_{13}y_{31} & y_{23}y_{32} & 0 & 0 & 0 & y_{13}^2+y_{23}^2 \end{bmatrix} \tag{A2}$$

Finally, the 3-antenna $\boldsymbol{J}^H\breve{\boldsymbol{r}}$ term becomes

$$\boldsymbol{J}^H\breve{\boldsymbol{r}} = \begin{bmatrix} \bar{y}_{12}r_{12} + \bar{y}_{13}r_{13} \\ \bar{y}_{21}r_{21} + \bar{y}_{23}r_{23} \\ \bar{y}_{31}r_{31} + \bar{y}_{32}r_{32} \\ y_{12}\bar{r}_{12} + y_{13}\bar{r}_{13} \\ y_{21}\bar{r}_{21} + y_{23}\bar{r}_{23} \\ y_{31}\bar{r}_{31} + y_{32}\bar{r}_{32} \end{bmatrix}. \tag{A3}$$

APPENDIX B: JACOBIAN-BASED OPTIMIZATION ALGORITHMS

This appendix documents the various standard least-squares optimization algorithms that are referenced in this paper:

B1 Algorithm SD (steepest descent)

(i) Start with a best guess for the parameter vector, $\boldsymbol{z}_0$;
(ii) At each step k, compute the residuals $\check{\boldsymbol{r}}_k$, and the Jacobian $\boldsymbol{J} = \boldsymbol{J}(\check{\boldsymbol{z}}_k)$;
(iii) Compute the parameter update as (note that due to redundancy, only the top half of the vector actually needs to be computed):

$$\delta\check{\boldsymbol{z}}_k = -\lambda \boldsymbol{J}^H \check{\boldsymbol{r}}_k,$$

where λ is some small value;
(iv) If not converged[4], set $\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \delta\boldsymbol{z}$, and go back to step (ii).

B2 Algorithm GN (Gauss-Newton)

(i) Start with a best guess for the parameter vector, $\boldsymbol{z}_0$;
(ii) At each step k, compute the residuals $\check{\boldsymbol{r}}_k$, and the Jacobian $\boldsymbol{J} = \boldsymbol{J}(\check{\boldsymbol{z}}_k)$;
(iii) Compute the parameter update $\delta\check{\boldsymbol{z}}$ using Eq. 4 with $\lambda = 0$ (note that only the top half of the vector actually needs to be computed);
(iv) If not converged, set $\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \delta\boldsymbol{z}$, and go back to step (ii).

B3 Algorithm LM (Levenberg-Marquardt)

Several variations of this exist, but a typical one is:

(i) Start with a best guess for the parameter vector, $\boldsymbol{z}_0$, and an initial value for the damping parameter, e.g. $\lambda = 1$;
(ii) At each step k, compute the residuals $\check{\boldsymbol{r}}_k$, and the cost function $\chi^2_k = ||\check{\boldsymbol{r}}_k||_F$.
(iii) If $\chi^2_k \geqslant \chi^2_{k-1}$ (unsuccessful step), reset $\boldsymbol{z}_k = \boldsymbol{z}_{k-1}$, and set $\lambda = \lambda K$ (where typically $K = 10$);
(iv) Otherwise (successful step) set $\lambda = \lambda/K$;
(v) Compute the Jacobian $\boldsymbol{J} = \boldsymbol{J}(\check{\boldsymbol{z}}_k)$;
(vi) Compute the parameter update $\delta\check{\boldsymbol{z}}_k$ using Eq. 4 or 5 (note that only the top half of the vector actually needs to be computed);
(vii) If not converged, set $\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \delta\boldsymbol{z}$, and go back to step (ii).

B4 Convergence

All of the above algortihms iterate to "convergence". One or more of the following convergence criteria may be implemented in each case:

- Parameter update smaller than some pre-defined threshold: $||\delta\boldsymbol{z}||_F < \delta_0$.
- Improvement to cost function smaller than some pre-defined threshold: $\chi^2_{k-1} - \chi^2_k < \epsilon_0$.
- Norm of the gradient smaller than some threshold: $||\boldsymbol{J}||_F < \gamma_0$.

[4] see below