Mon. Not. R. Astron. Soc. **000**, 1–17 (2014) Printed 16 January 2015 (MN LATEX style file v2.2)

Radio interferometric gain calibration as a complex optimization problem

O.M. Smirnov[12⋆], C. Tasse[31]
[1] *Department of Physics and Electronics, Rhodes University, PO Box 94, Grahamstown, 6140 South Africa*
[2] *SKA South Africa, 3rd Floor, The Park, Park Road, Pinelands, 7405 South Africa*
[3] *GEPI, Observatoire de Paris, CNRS, Université Paris Diderot, 5 place Jules Janssen, 92190 Meudon, France*

in original form 2014 October 30

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

INTRODUCTION

In radio interferometry, gain calibration consists of solving for the unknown complex antenna gains, using a known (prior, or iteratively constructed) model of the sky. Traditional (second generation, or 2GC) calibration employs an instrumental model with a single direction-independent (DI) gain term (which can be a scalar complex gain, or 2×2 complex-valued *Jones matrix*) per antenna, per some time/frequency interval. Third-generation (3GC) calibration also addresses direction-dependent (DD) effects, which can be represented by independently solvable DD gain terms, or by some parameterized instrumental model (e.g. primary beams, pointing offsets, ionospheric screens). Different approaches to this have been proposed and implemented, mostly in the framework of the radio interferometry measurement equation (RIME, see Hamaker et al. 1996); Smirnov (2011a,b,c) provides a recent overview. In this work we will restrict ourselves specifically to calibration of the DI and DD gains terms (the latter in the sense of being solved independently per direction).

Gain calibration is a non-linear least squares (NLLS) problem, since the noise on observed visibilities is almost always Gaussian (though other treatments have been proposed by Kazemi & Yatawatta 2013). Traditional approaches to NLLS problems involve various gradient-based techniques (for an overview, see Madsen et al. 2004), such as Gauss-Newton (GN) and Levenberg-Marquardt (LM). These have been restricted to functions of real variables, since complex differentiation can be defined in only a very restricted sense (in particular, $\partial\bar{z}/\partial z$ does not exist in the usual definition). Gains in radio interferometry are complex variables: the traditional way out of this conundrum has been to recast the complex NLLS problem as a real problem by treating the real and imaginary parts of the gains as independent real variables.

Recent developments in optimization theory (Kreutz-Delgado 2009; Laurent et al. 2012) have shown that using a formalism called the *Wirtinger complex derivative* (Wirtinger 1927) allows for a mathematically robust definition of a complex gradient operator. This leads to the con-

⋆ E-mail: o.smirnov@ru.ac.za

struction of a *complex Jacobian* $\boldsymbol{J}$, which in turn allows for traditional NLLS algorithms to be directly applied to the complex variable case. We summarize these developments and introduce basic notation in Sect. 1. In Sect. 2, we follow on from Tasse (2014) to apply this theory to the RIME, and derive complex Jacobians for (unpolarized) DI and DD gain calibration.

In principle, the use of Wirtinger calculus and complex Jacobians ultimately results in the same system of LS equations as the real/imaginary approach. It does offer two important advantages: (i) equations with complex variables are more compact, and are more natural to derive and analyze than their real/imaginary counterparts, and (ii) the structure of the complex Jacobian can yield new and valuable insights into the problem. This is graphically illustrated in Fig. 1 (in fact, this figure may be considered the central insight of this paper). Methods such as GN and LM hinge around a large matrix – $\boldsymbol{J}^H\boldsymbol{J}$ – with dimensions corresponding to the number of free parameters; construction and/or inversion of this matrix is often the dominant algorithmic cost. If $\boldsymbol{J}^H\boldsymbol{J}$ can be treated as (perhaps approximately) sparse, these costs can be reduced, often drastically. Figure 1 shows the structure of an example $\boldsymbol{J}^H\boldsymbol{J}$ matrix for a DD gain calibration problem. The left column row shows versions of $\boldsymbol{J}^H\boldsymbol{J}$ constructed via the real/imaginary approach, for four different orderings of the solvable parameters. None of the orderings yield a matrix that is particularly sparse or easily invertable. The right column shows a complex $\boldsymbol{J}^H\boldsymbol{J}$ for the same orderings. Panel (f) reveals sparsity that is not apparent in the real/imaginary approach. This sparsity forms the basis of a new fast DD calibration algorithm discussed later in the paper.

In Sect. 3, we show that different algorithms may be derived by combining different sparse approximations to $\boldsymbol{J}^H\boldsymbol{J}$ with conventional GN and LM methods. In particular, we show that StefCal, a fast DI calibration algorithm recently proposed by Salvini & Wijnholds (2014a), can be straightforwardly derived from a diagonal approximation to a complex $\boldsymbol{J}^H\boldsymbol{J}$. We show that the complex Jacobian approach naturally extends to the DD case, and that other sparse approximations yield a whole family of DD calibration algorithms with different scaling properties. One such algorithm, CohJones (Tasse 2014), has been implemented and successfully applied to simulated LOFAR data: this is discussed in Sect. 6.

In Sect. 4 we extend this approach to the fully polarized case, by developing a Wirtinger-like operator calculus in which the polarization problem can be formulated succinctly. This naturally yields fully polarized counterparts to the calibration algorithms defined previously. In Sect. 5, we discuss other algortihmic variations, and make connections to older DD calibration techniques such as peeling (Noordam 2004).

While the scope of this work is restricted to LS solutions to the DI and DD gain calibration problem, the potential applicability of complex optimization to radio interferometry is perhaps broader. We will return to this in the conclusions.

Table 1. Notation and frequently used symbols

x	scalar value x
$\bar{x}$	complex conjugate
$\boldsymbol{x}$	vector $\boldsymbol{x}$
$\boldsymbol{X}$	matrix $\boldsymbol{X}$
$\mathbf{X}$	vector of 2×2 matrices $\mathbf{X} = [\boldsymbol{X}_i]$ (Sect. 4)
$\mathbb{R}$	space of real numbers
$\mathbb{C}$	space of complex numbers
$\mathbb{I}$	identity matrix
$\operatorname{diag}\boldsymbol{x}$	diagonal matrix formed from $\boldsymbol{x}$
$\|\|\cdot\|\|_F$	Frobenius norm
$(\cdot)^T$	transpose
$(\cdot)^H$	Hermitian transpose
$\otimes$	outer product a.k.a. Kronecker product
$\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}}$	element-by-element complex conjugate of $\boldsymbol{x}$, $\boldsymbol{X}$
$\check{\boldsymbol{x}}, \check{\mathbf{X}}$	augmented vectors $\check{\boldsymbol{x}} = \begin{bmatrix}\boldsymbol{x}\\ \bar{\boldsymbol{x}}\end{bmatrix}$, $\check{\mathbf{X}} = \begin{bmatrix}\boldsymbol{X}_i\\ \boldsymbol{X}_i^H\end{bmatrix}$
$\boldsymbol{X}_{\mathrm{U}}$	upper half of matrix $\boldsymbol{X}$
$\boldsymbol{X}_{\mathrm{L}}, \boldsymbol{X}_{\mathrm{R}}$	left, right half of matrix $\boldsymbol{X}$
$\boldsymbol{X}_{\mathrm{UL}}$	upper left quadrant of matrix $\boldsymbol{X}$
	order of operations is $\boldsymbol{X}_{\mathrm{U}}^Y = \boldsymbol{X}_{\mathrm{U}}{}^Y = (\boldsymbol{X}_{\mathrm{U}})^Y$, or $\boldsymbol{X}^Y{}_{\mathrm{U}} = (\boldsymbol{X}^Y)_{\mathrm{U}}$
$\boldsymbol{d}, \boldsymbol{v}, \boldsymbol{r}, \boldsymbol{g}, \boldsymbol{m}$	data, model, residuals, gains, sky coherency
$(\cdot)_k$	value associated with iteration k
$(\cdot)_{p,k}$	value associated with antenna p, iteration k
$(\cdot)^{(d)}$	value associated with direction d
$\boldsymbol{W}$	matrix of weights
$\boldsymbol{J}_k, \boldsymbol{J}_{k^*}$	partial, conjugate partial Jacobian at iteration k
$\boldsymbol{J}$	full complex Jacobian
$\boldsymbol{H}, \tilde{\boldsymbol{H}}$	$\boldsymbol{J}^H\boldsymbol{J}$ and its approximation
$\operatorname{vec}\boldsymbol{X}$	vectorization operator
$\mathcal{R}_A$	right-multiply by $\boldsymbol{A}$ operator (Sect. 4)
$\mathcal{L}_A$	left-multiply by $\boldsymbol{A}$ operator (Sect. 4)
δ^i_j	Kronecker delta symbol
$\searrow, \nearrow, \downarrow$	repeated matrix block

1 WIRTINGER CALCULUS & COMPLEX LEAST-SQUARES

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 $\mathbb{R}^{2n}$.

Kreutz-Delgado (2009) and Laurent et al. (2012) propose an alternative approach to the problem based on Wirtinger (1927) 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*

$$\frac{\partial}{\partial z} = \frac{1}{2}\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right), \quad \frac{\partial}{\partial \bar{z}} = \frac{1}{2}\left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right), \tag{1.1}$$

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

$$\frac{\partial \bar{z}}{\partial z} = \frac{\partial z}{\partial \bar{z}} = 0, \tag{1.2}$$

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

$$\frac{\partial}{\partial^C \boldsymbol{z}} = \left[\frac{\partial}{\partial \boldsymbol{z}}, \frac{\partial}{\partial \bar{\boldsymbol{z}}}\right] = \left[\frac{\partial}{\partial z_1}, \ldots, \frac{\partial}{\partial z_n}, \frac{\partial}{\partial \bar{z}_1}, \ldots, \frac{\partial}{\partial \bar{z}_n}\right], \tag{1.3}$$

from which definitions of the complex Jacobian and complex Hessians naturally follow. The authors then show that vari-

ous optimization techniques developed for real functions can be reformulated using complex Jacobians and Hessians, and applied to the complex optimization problem. In particular, they generalize the Gauss-Newton (GN) and Levenberg-Marquardt (LM) methods for solving the non-linear least squares (NLLS) problem[1]

$$\min_{\boldsymbol{z}} ||\boldsymbol{r}(\boldsymbol{z},\bar{\boldsymbol{z}})||_F, \quad \text{or} \quad \min_{\boldsymbol{z}} ||\boldsymbol{d}-\boldsymbol{v}(\boldsymbol{z},\bar{\boldsymbol{z}})||_F \tag{1.4}$$

where $\boldsymbol{r},\boldsymbol{d},\boldsymbol{v}$ have values in $\mathbb{C}^m$, and $||\cdot||_F$ is the Frobenius norm. The latter form refers to LS fitting of the parameterized model $\boldsymbol{v}$ to observed data $\boldsymbol{d}$, and is the preferred formulation in the context of radio interferometry.

Complex NLLS is implemented as follows. Let us formally treat $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$ as independent variables, define an *augmented parameter vector* containing both,

$$\check{\boldsymbol{z}} = \begin{bmatrix}\boldsymbol{z}\\ \bar{\boldsymbol{z}}\end{bmatrix} \tag{1.5}$$

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

$$\boldsymbol{J}_k = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{z}}(\check{\boldsymbol{z}}_k),\ \boldsymbol{J}_{k^*} = \frac{\partial \boldsymbol{v}}{\partial \bar{\boldsymbol{z}}}(\check{\boldsymbol{z}}_k),\ \check{\boldsymbol{r}}_k = \begin{bmatrix}\boldsymbol{r}(\check{\boldsymbol{z}}_k)\\ \bar{\boldsymbol{r}}(\check{\boldsymbol{z}}_k)\end{bmatrix} \tag{1.6}$$

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

$$\boldsymbol{J} = \begin{bmatrix}\boldsymbol{J}_k & \boldsymbol{J}_{k^*}\\ \bar{\boldsymbol{J}}_{k^*} & \bar{\boldsymbol{J}}_k\end{bmatrix}, \tag{1.7}$$

with the bottom two blocks being element-by-element conjugated versions of the top two. Note the use of $\bar{\boldsymbol{J}}$ to indicate element-by-element conjugation – not to be confused with the Hermitian conjugate which we'll invoke later. $\boldsymbol{J}$ is a $2m\times 2n$ matrix. The GN update step is defined as

$$\delta\check{\boldsymbol{z}} = \begin{bmatrix}\delta\boldsymbol{z}\\ \delta\bar{\boldsymbol{z}}\end{bmatrix} = (\boldsymbol{J}^H\boldsymbol{J})^{-1}\boldsymbol{J}^H\check{\boldsymbol{r}}_k, \tag{1.8}$$

The LM approach is similar, but introduces a *damping parameter* λ:

$$\delta\check{\boldsymbol{z}} = \begin{bmatrix}\delta\boldsymbol{z}\\ \delta\bar{\boldsymbol{z}}\end{bmatrix} = (\boldsymbol{J}^H\boldsymbol{J}+\lambda\boldsymbol{D})^{-1}\boldsymbol{J}^H\check{\boldsymbol{r}}_k, \tag{1.9}$$

where $\boldsymbol{D}$ is the diagonalized version of $\boldsymbol{J}^H\boldsymbol{J}$. With $\lambda=0$ this becomes equivalent to GN, with $\lambda\to\infty$ this corresponds to steepest descent (SD) with ever smaller steps.

Note that while $\delta\boldsymbol{z}$ and $\delta\bar{\boldsymbol{z}}$ are formally computed independently, the structure of the equations is symmetric (since the function being minimized – the Frobenius norm – is real and symmetric w.r.t. $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$), which ensures 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 Eqs. 1.8 and 1.9 yield 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.

2 SCALAR (UNPOLARIZED) CALIBRATION

In this section we will apply the formalism above to the scalar case, i.e. that of unpolarized calibration. This will then be extended to the fully polarized case in Sect. 4.

2.1 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[3]

$$g_p m_{pq}\bar{g}_q + n_{pq}, \tag{2.1}$$

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:

$$\min_{\boldsymbol{g}} \sum_{pq} |r_{pq}|^2, \quad r_{pq} = d_{pq} - g_p m_{pq}\bar{g}_q, \tag{2.2}$$

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. 1.6). This is of shape $N_{\rm bl}\times N_{\rm ant}$. Using the Wirtinger derivative, we can write the partial Jacobian in terms of its value at row $[pq]$ and column j as

$$[\boldsymbol{J}_k]_{[pq],j} = \begin{cases} m_{pq}\bar{g}_q, & j=p,\\ 0, & \text{otherwise.}\end{cases} \tag{2.3}$$

[1] It should be stressed that Wirtinger calculus can be applied to a broader range of optimization problems than just LS.

[2] Laurent et al. (2012) define the Jacobian via $\partial\boldsymbol{r}$ rather than $\partial\boldsymbol{v}$. This yields a Jacobian of the opposite sign, and introduces a minus sign into Eqs. 1.8 and 1.9. In this paper we use the $\partial\boldsymbol{v}$ convention, as is more common in the context of radio interferometric calibration.

[3] In principle, the *autocorrelation* terms pp, corresponding to the total power in the field, are also measured, and may be incorporated into the equations here. It is, however, common practice to omit autocorrelations from the interferometric calibration problem due to their much higher noise, as well as technical difficulties in modelling the total intensity contribution. The derivations below are equally valid for $p\leqslant q$; we use $p<q$ for consistency with practice.

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

$$J_k = \overbrace{\left[m_{pq}\bar{g}_q\delta_p^j\right]}^{j=1\ldots N_{\rm ant}}\Big\}_{[pq]=1\ldots N_{\rm bl}\ (p<q)} \tag{2.4}$$

Likewise, the conjugate partial Jacobian J_{k^*} may be written as

$$J_{k^*} = \overbrace{\left[g_p m_{pq}\delta_q^j\right]}^{j=1\ldots N_{\rm ant}}\Big\}_{[pq]=1\ldots N_{\rm bl}\ (p<q)} \tag{2.5}$$

A specific example is provided in Appendix A. The full complex Jacobian (Eq. 1.7) then becomes, in block matrix notation,

$$\boldsymbol{J} = \begin{bmatrix} \overbrace{m_{pq}\bar{g}_q\delta_p^j}^{j=1\ldots N_{\rm ant}} & \overbrace{g_p m_{pq}\delta_q^j}^{j=1\ldots N_{\rm ant}} \\ \bar{m}_{pq}\bar{g}_p\delta_q^j & g_q\bar{m}_{pq}\delta_p^j \end{bmatrix} \begin{matrix} \Big\}_{[pq]=1\ldots N_{\rm bl}\ (p<q)} \\ \Big\}_{[pq]=1\ldots N_{\rm bl}\ (p<q)} \end{matrix} \tag{2.6}$$

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 $(q>p)$:

$$\breve{\boldsymbol{r}} = \begin{bmatrix} r_{pq} \\ \bar{r}_{pq} \end{bmatrix} = \begin{bmatrix} d_{pq} - g_p m_{pq}\bar{g}_q \\ \bar{d}_{pq} - \bar{g}_p\bar{m}_{pq}g_q \end{bmatrix} = \begin{bmatrix} r_{pq} \\ r_{qp} \end{bmatrix} \begin{matrix} \Big\}_{[pq]=1\ldots N_{\rm bl}\ (p<q)} \\ \Big\}_{[pq]=1\ldots N_{\rm bl}\ (p<q)} \end{matrix} \tag{2.7}$$

as does the bottom half of $\boldsymbol{J}$ in Eq. 2.6. 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. 1.9. 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:

$$\boldsymbol{J} = \begin{bmatrix} \overbrace{m_{pq}\bar{g}_q\delta_p^j}^{j=1\ldots N_{\rm ant}} & \overbrace{g_p m_{pq}\delta_q^j}^{j=1\ldots N_{\rm ant}} \end{bmatrix},\ \breve{\boldsymbol{r}} = \begin{bmatrix} r_{pq} \end{bmatrix}\ \Big\}_{[pq]=1\ldots 2N_{\rm bl}} \tag{2.8}$$

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 A2 in the Appendix provides an example of $\boldsymbol{J}$ for the 3-antenna case. For brevity, let us define the shorthand

$$y_{pq} = m_{pq}\bar{g}_q. \tag{2.9}$$

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:

$$\boldsymbol{J}^H\boldsymbol{J} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^H & \boldsymbol{A} \end{bmatrix} \tag{2.10}$$

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

$$\begin{aligned} A_{ij} = \sum_{pq} \bar{y}_{pq}y_{pq}\delta_p^i\delta_p^j &= \begin{cases} \sum\limits_{q\neq i}|y_{iq}^2|, & i=j \\ 0, & i\neq j \end{cases} \\ B_{ij} = \sum_{pq} \bar{y}_{pq}\bar{y}_{qp}\delta_p^i\delta_q^j &= \begin{cases} \bar{y}_{ij}\bar{y}_{ji}, & i\neq j \\ 0, & i=j \end{cases} \\ C_{ij} = \sum_{pq} y_{qp}y_{pq}\delta_q^i\delta_p^j &= \bar{B}_{ij} \\ D_{ij} = \sum_{pq} y_{pq}\bar{y}_{pq}\delta_q^i\delta_q^j &= A_{ij} \end{aligned} \tag{2.11}$$

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

$$\boldsymbol{J}^H\boldsymbol{J} = \left[\begin{array}{c|c} \operatorname{diag}\sum\limits_{q\neq i}|y_{iq}^2| & \begin{cases} \bar{y}_{ij}\bar{y}_{ji}, & i\neq j \\ 0, & i=j \end{cases} \\ \hline \begin{cases} y_{ij}y_{ji}, & i\neq j \\ 0, & i=j \end{cases} & \operatorname{diag}\sum\limits_{q\neq i}|y_{iq}^2| \end{array}\right] \tag{2.12}$$

Equation A3 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. 1.9) 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:

$$\boldsymbol{J}^H\breve{\boldsymbol{r}} = \begin{bmatrix} \sum\limits_{pq}\bar{y}_{pq}r_{pq}\delta_p^i \\ \sum\limits_{pq}y_{qp}r_{pq}\delta_q^i \end{bmatrix} = \begin{bmatrix} \sum\limits_{q\neq i}\bar{y}_{iq}r_{iq} \\ \sum\limits_{q\neq i}y_{iq}\bar{r}_{iq} \end{bmatrix} \begin{matrix} \Big\}_{i=1\ldots N_{\rm ant}} \\ \Big\}_{i=1\ldots N_{\rm ant}} \end{matrix} \tag{2.13}$$

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:

$$\boldsymbol{J}^H\breve{\boldsymbol{r}} = \begin{bmatrix} \boldsymbol{c} \\ \bar{\boldsymbol{c}} \end{bmatrix},\quad c_i = \sum_{q\neq i}\bar{y}_{iq}r_{iq}. \tag{2.14}$$

2.2 Computing the parameter update

Due to the structure of the RIME, we have a particularly elegant way of computing the GN update step. 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 $[pq]$ $(p\neq q)$:

$$\breve{\boldsymbol{d}} = [d_{pq}],\quad \breve{\boldsymbol{v}} = [g_p m_{pq}\bar{g}_q],\quad \breve{\boldsymbol{r}} = \breve{\boldsymbol{d}} - \breve{\boldsymbol{v}} \tag{2.15}$$

As noted by Tasse (2014), we have the wonderful property that

$$\breve{\boldsymbol{v}} = \boldsymbol{J}_{\rm L}\boldsymbol{g} = \frac{1}{2}\boldsymbol{J}\breve{\boldsymbol{g}},\quad \text{where } \breve{\boldsymbol{g}} = \begin{bmatrix} \boldsymbol{g} \\ \bar{\boldsymbol{g}} \end{bmatrix}, \tag{2.16}$$

(where $\boldsymbol{X}_{\rm L}$ designates the left half of matrix $\boldsymbol{X}$ – see Table 1 for a summary of notation), which basically comes about due to the RIME being bilinear with respect to $\boldsymbol{g}$ and $\bar{\boldsymbol{g}}$. Substituting this into the GN update step, and noting that $\boldsymbol{X}(\boldsymbol{Y}_{\rm L}) = (\boldsymbol{X}\boldsymbol{Y})_{\rm L}$, we have

$$\begin{bmatrix} \delta\boldsymbol{g} \\ \delta\bar{\boldsymbol{g}} \end{bmatrix} = (\boldsymbol{J}^H\boldsymbol{J})^{-1}\boldsymbol{J}^H(\breve{\boldsymbol{d}} - \boldsymbol{J}_{\rm L}\boldsymbol{g}) = (\boldsymbol{J}^H\boldsymbol{J})^{-1}\boldsymbol{J}^H\breve{\boldsymbol{d}} - \boldsymbol{g}. \tag{2.17}$$

Consequently, the updated gain values at each iteration can be derived directly from the data, thus obviating the need for computing residuals. Additionally, since the bottom half

of the equations is simply the conjugate of the top, we only need to evaluate the top half:

$$\boldsymbol{g}_{k+1} = \boldsymbol{g}_k + \delta\boldsymbol{g} = (\boldsymbol{J}^H\boldsymbol{J})^{-1}{}_{\mathrm{U}}\boldsymbol{J}^H\check{\boldsymbol{d}}, \tag{2.18}$$

where $\boldsymbol{X}_{\mathrm{U}}$ designates the upper half of matrix $\boldsymbol{X}$.

The derivation above assumes an exact inversion of $\boldsymbol{H} = \boldsymbol{J}^H\boldsymbol{J}$. In practice, this large matrix can be costly to invert, so the algorithms below will substitute it with some cheaper-to-invert approximation $\tilde{\boldsymbol{H}}$. Using the approximate matrix in the GN update equation, we find instead that

$$\boldsymbol{g}_{k+1} = \tilde{\boldsymbol{H}}^{-1}{}_{\mathrm{U}}\boldsymbol{J}^H\check{\boldsymbol{d}} + (\mathbb{I} - \tilde{\boldsymbol{H}}^{-1}{}_{\mathrm{U}}\boldsymbol{H}_{\mathrm{L}})\boldsymbol{g}_k, \tag{2.19}$$

which means that when an approximate $\boldsymbol{J}^H\boldsymbol{J}$ is in use, the shortcut of Eq. 2.18 only applies when

$$\tilde{\boldsymbol{H}}^{-1}{}_{\mathrm{U}}\boldsymbol{H}_{\mathrm{L}} = \mathbb{I}. \tag{2.20}$$

We will see examples of both conditions below, so it is worth stressing the difference: Eq. 2.18 allows us to compute updated solutions directly from the data vector, bypassing the residuals. This is a substantial computational shortcut, however, when an approximate inverse for $\boldsymbol{H}$ is in use, it does not necessarily apply (or at least is not exact). Under the condition of Eq. 2.20, however, such a shortcut is exact.

2.3 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. 2.1 can then be re-written as

$$\min_{\boldsymbol{g}} \sum_{pqs} |r_{pqs}|^2, \quad r_{pqs} = d_{pqs} - g_p m_{pqs}\bar{g}_q, \tag{2.21}$$

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 $2N_{\mathrm{bl}} \times 2N_{\mathrm{ant}}$, the Jacobian will have a shape of $2N_{\mathrm{bl}}N_s \times 2N_{\mathrm{ant}}$, and the residual vector will have a length of $2N_{\mathrm{bl}}N_s$. In deriving the $\boldsymbol{J}^H\boldsymbol{J}$ term, the sums in Eq. 2.10 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:

$$\boldsymbol{J}^H\boldsymbol{J} = \begin{bmatrix} \operatorname{diag} \sum\limits_{q\neq i,s} |y_{iqs}^2| & \Big| \nearrow^H \\ \hdashline \begin{cases} \sum\limits_s y_{ijs}y_{jis}, & i\neq j \\ 0, & i=j \end{cases} & \Big| \searrow \end{bmatrix}, \tag{2.22}$$

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. 2.10). Likewise, the $\boldsymbol{J}^H\check{\boldsymbol{r}}$ term can be written as:

$$\boldsymbol{J}^H\check{\boldsymbol{r}} = \begin{bmatrix} \sum\limits_{q\neq i,s} \bar{y}_{iqs}r_{iqs} \\ \hdashline \downarrow^H \end{bmatrix}. \tag{2.23}$$

2.4 Weighting

Although Laurent et al. (2012) do not mention this explicitly, it is straightforward to incorporate weights into the complex LS problem. Equation 1.4 is reformulated as

$$\min_{\boldsymbol{z}} ||\boldsymbol{W}\check{\boldsymbol{r}}(\boldsymbol{z}, \bar{\boldsymbol{z}})||_F, \tag{2.24}$$

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

$$\delta\check{\boldsymbol{z}} = (\boldsymbol{J}^H\boldsymbol{W}\boldsymbol{J} + \lambda\mathbb{I})^{-1}\boldsymbol{J}^H\boldsymbol{W}\check{\boldsymbol{r}}_k. \tag{2.25}$$

Adding weights to Eqs. 2.22 and 2.23, we arrive at the following:

$$\boldsymbol{J}^H\boldsymbol{W}\boldsymbol{J} = \begin{bmatrix} \operatorname{diag} \sum\limits_{q\neq i,s} w_{iqs}|y_{iqs}^2| & \Big| \nearrow^H \\ \hdashline \begin{cases} \sum\limits_s w_{ijs}y_{ijs}y_{jis}, & i\neq j \\ 0, & i=j \end{cases} & \Big| \searrow \end{bmatrix} \tag{2.26}$$

$$\boldsymbol{J}^H\boldsymbol{W}\check{\boldsymbol{r}} = \begin{bmatrix} \sum\limits_{q\neq i,s} w_{iqs}\bar{y}_{iqs}r_{iqs} \\ \hdashline \downarrow^H \end{bmatrix}. \tag{2.27}$$

2.5 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_{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:

$$\min_{\boldsymbol{g}} \sum_{pqs} |r_{pqs}|^2, \quad r_{pqs} = d_{pqs} - \sum_{d=1}^{N_{\mathrm{dir}}} g_p^{(d)} m_{pqs}^{(d)} \bar{g}_q^{(d)}. \tag{2.28}$$

It's obvious that the Jacobian corresponding to this problem is very similar to the one in Eq. 2.8, but instead of having shape $2N_{\mathrm{bl}} \times 2N_{\mathrm{ant}}$, this will have a shape of $2N_{\mathrm{bl}}N_s \times 2N_{\mathrm{ant}}N_{\mathrm{dir}}$. We now treat $[pqs]$ and $[jd]$ as compound indices:

$$\boldsymbol{J} = \left\{ \begin{bmatrix} \overbrace{m_{pqs}^{(d)}\bar{g}_q^{(d)}\delta_p^j}^{\substack{j=1\ldots N_{\mathrm{ant}} \\ d=1\ldots N_{\mathrm{dir}}}} & \overbrace{g_p^{(d)}m_{pqs}^{(d)}\delta_q^j}^{\substack{j=1\ldots N_{\mathrm{ant}} \\ d=1\ldots N_{\mathrm{dir}}}} \end{bmatrix} \right\} \begin{matrix} [pq]=1\ldots 2N_{\mathrm{bl}}\ (p\neq q) \\ s=1\ldots N_s \end{matrix} \tag{2.29}$$

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 $\check{\boldsymbol{g}}$) is completely up to us.

Consider now the $\boldsymbol{J}^H\boldsymbol{J}$ product. This will consist of 2×2 blocks, each of shape $[N_{\mathrm{ant}}N_{\mathrm{dir}}]^2$. 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:

$$\boldsymbol{J}^H\boldsymbol{J} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B}^H \\ \boldsymbol{B} & \boldsymbol{A} \end{bmatrix} = \left[\begin{array}{cc|c} \multicolumn{2}{c|}{\delta^i_j \sum_{q\neq i,s} \bar{y}^{(c)}_{iqs} y^{(d)}_{iqs}} & \nearrow^H \\ \hdashline \sum_s y^{(c)}_{jis} y^{(d)}_{ijs}, & i\neq j & \\ 0 & i=j & \searrow \end{array}\right], \quad (2.30)$$

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:

$$\boldsymbol{J}^H\breve{\boldsymbol{r}} = \begin{bmatrix} \boldsymbol{c} \\ \bar{\boldsymbol{c}} \end{bmatrix}, \quad c_{jd} = \sum_{q\neq j,s} \bar{y}^{(d)}_{jqs} r_{jqs}. \quad (2.31)$$

Finally, let us note that the property of Eq. 2.16 also holds for the DD case. It is easy to see that

$$\breve{\boldsymbol{v}} = \left[\sum_{d=1}^{N_{\rm dir}} g^{(d)}_p m^{(d)}_{pqs} \bar{g}^{(d)}_q\right] = \boldsymbol{J}_{\rm L}\breve{\boldsymbol{g}}. \quad (2.32)$$

Consequently, the shortcut of Eq. 2.18 also applies.

3 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. 2.12+2.14, 2.22+2.23, 2.26+2.27 or 2.30+2.31 into one the algorithms defined in Appendix C. 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 $O(N^3_{\rm ant})$ or $O(N^3_{\rm ant}N^3_{\rm dir})$ cost of inversion.

In this section we investigate approaches to simplifying the inversion problem by approximating $\boldsymbol{H} = \boldsymbol{J}^H\boldsymbol{J}$ by some form of (block-)diagonal matrix $\tilde{\boldsymbol{H}}$. 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.

3.1 Diagonal approximation and StefCal

Let us first consider the DI case. The structure of $\boldsymbol{J}^H\boldsymbol{J}$ in Eq. 2.12 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(f). 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):

$$\tilde{\boldsymbol{H}} = \begin{bmatrix} \boldsymbol{A} & 0 \\ 0 & \boldsymbol{A} \end{bmatrix} \quad (3.1)$$

This makes the costs of matrix inversion negligible – $O(N_{\rm ant})$ operations, as compared to the $O(N^2_{\rm ant})$ 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 and using Eq. 2.19, we find the following expression for the GN update step:

$$\boldsymbol{g}_{k+1} = \tilde{\boldsymbol{H}}^{-1}_{\rm UL}{\boldsymbol{J}_{\rm L}}^H\breve{\boldsymbol{d}}, \quad (3.2)$$

where $\boldsymbol{X}_{\rm UL}$ designates the top left quadrant of matrix $\boldsymbol{X}$. Note that the condition of Eq. 2.20 is met: ${\tilde{\boldsymbol{H}}^{-1}}_{\rm U}\boldsymbol{H}_{\rm L} = \boldsymbol{A}^{-1}\boldsymbol{A} = \mathbb{I}$, i.e. the GN update can be written in terms of $\breve{\boldsymbol{d}}$. This comes about due to (i) the off-diagonal blocks of $\tilde{\boldsymbol{H}}$ being null, which masks out the bottom half of $\boldsymbol{H}_{\rm L}$, and (ii) the on-diagonal blocks of $\tilde{\boldsymbol{H}}$ being an exact inverse. In other algorithms suggested below, the second condition particularly is not the case.

The per-element expression, in the diagonal approximation, is

$$g_{p,k+1} = \big(\sum_{q\neq p} \bar{y}_{pq}y_{pq}\big)^{-1} \sum_{q\neq p} \bar{y}_{pq}d_{pq}. \quad (3.3)$$

Equation 3.3 is identical to the update step proposed by Hamaker (2000), and later adopted by Mitchell et al. (2008) for MWA calibration, and independently derived by Salvini & Wijnholds (2014a) for the StefCal algorithm. Note that these authors arrive at the result from a different direction, by treating Eq. 2.2 as a function of $\boldsymbol{g}$ only, and completely ignoring the conjugate term. The resulting complex Jacobian (Eq. 2.6) then has null off-diagonal blocks, and $\boldsymbol{J}^H\boldsymbol{J}$ becomes diagonal.

Interestingly, applying the same idea to LM optimization (Eq. 1.9), and remembering that $\tilde{\boldsymbol{H}}$ is diagonal, we can derive the following update equation instead:

$$\boldsymbol{g}_{k+1} = \frac{\lambda}{1+\lambda}\boldsymbol{g}_k + \frac{1}{1+\lambda}\tilde{\boldsymbol{H}}^{-1}_{\rm UL}{\boldsymbol{J}_{\rm L}}^H\breve{\boldsymbol{d}}, \quad (3.4)$$

which for $\lambda = 1$ essentially becomes the basic average-update step of StefCal. We should note that Salvini & Wijnholds (2014a) empirically find better convergence when Eq. 3.2 is employed for odd k, and Eq. 3.4 for even k. In terms of the framework defined here, the basic StefCal algorithm can be succinctly described as *complex optimization with a diagonally-approximated $\boldsymbol{J}^H\boldsymbol{J}$, using GN for the odd steps, and LM ($\lambda = 1$) for the even steps.*

Establishing this equivalence is very useful for our purposes, since the convergence properties of StefCal have been thoroughly explored by Salvini & Wijnholds (2014a), and we can therefore hope to apply these lessons here. In particular, these authors have shown that a direct application of GN produces very slow (oscillating) convergence, whereas combining GN and LM 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. 3.3 is embarassingly parallel, in the sense that the update for each antenna is computed entirely independently.

3.2 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

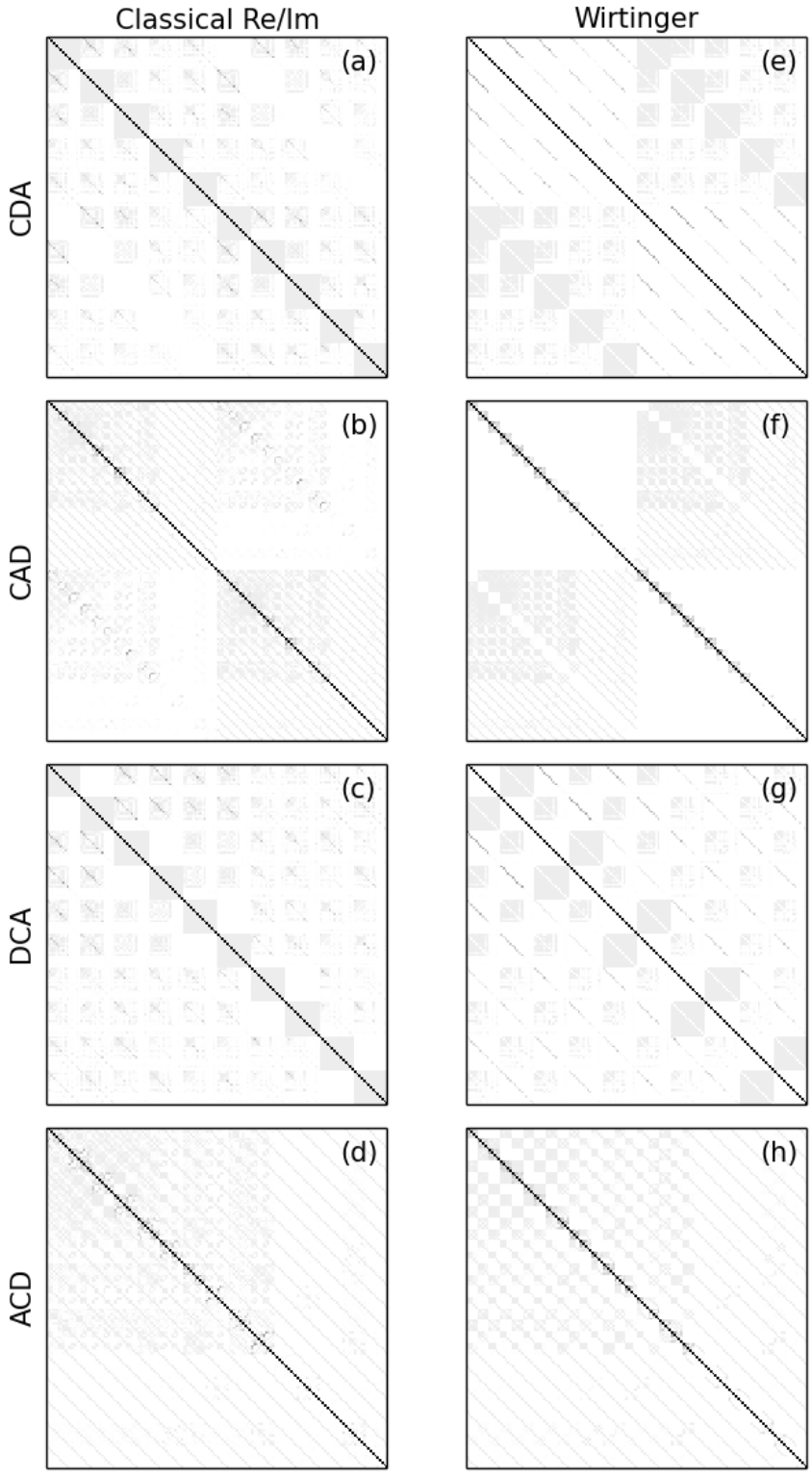

Figure 1. A graphical representation of $\boldsymbol{J}^H\boldsymbol{J}$ for a case of 40 antennas and 5 directions. Each pixel represents the amplitude of a single matrix element. The left column (a–d) shows conventional real-only Jacobians constructed by taking the partial derivatives w.r.t. the real and imaginary parts of the gains. The ordering of the parameters is (a) real/imaginary major, direction, antenna minor (i.e. antenna changes fastest); (b) real/imaginary, antenna, direction; (c) direction, real/imaginary, antenna; (d) antenna, real/imaginary, direction. The right column (e–h) shows full complex Jacobians with similar parameter ordering (direct/conjugate instead of real/imaginary). Note that panel (f) can also be taken to represent the direction-independent case, if we imagine each 5×5 block as one pixel.

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 $\check{\boldsymbol{g}}$:

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

$$[g_1^{(1)} \dots g_{N_{\rm ant}}^{(1)}, g_1^{(2)} \dots g_{N_{\rm ant}}^{(2)}, g_1^{(3)} \dots g_{N_{\rm ant}}^{(N_{\rm dir})}, \bar{g}_1^{(1)} \dots \bar{g}_{N_{\rm ant}}^{(1)} \dots]^T \tag{3.5}$$

- conjugate, antenna, direction (CAD):

$$[g_1^{(1)} \dots g_1^{(N_{\rm dir})}, g_2^{(1)} \dots g_2^{(N_{\rm dir})}, g_3^{(1)} \dots g_{N_{\rm ant}}^{(N_{\rm dir})}, \bar{g}_1^{(1)} \dots]^T \tag{3.6}$$

- direction, conjugate, antenna (DCA):

$$[g_1^{(1)} \dots g_{N_{\rm ant}}^{(1)}, \bar{g}_1^{(1)} \dots \bar{g}_{N_{\rm ant}}^{(1)}, g_1^{(2)} \dots g_{N_{\rm ant}}^{(2)}, \bar{g}_1^{(2)} \dots]^T \tag{3.7}$$

- antenna, conjugate, direction (ACD):

$$[g_1^{(1)} \dots g_1^{(N_{\rm dir})}, \bar{g}_1^{(1)} \dots \bar{g}_1^{(N_{\rm dir})}, g_2^{(1)} \dots g_2^{(N_{\rm dir})}, \bar{g}_2^{(1)} \dots]^T \tag{3.8}$$

Figure 1(e–h) graphically illustrates the structure of the Jacobian under these orderings.

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 $\check{\boldsymbol{g}}$, 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.

3.2.1 DCA: separating by direction

Let us first consider the DCA ordering (Fig. 1g). The $\boldsymbol{J}^H\boldsymbol{J}$ term can be split into $N_{\rm dir} \times N_{\rm dir}$ blocks:

$$\boldsymbol{J}^H\boldsymbol{J} = \begin{bmatrix} \mathcal{J}_1^1 & \dots & \mathcal{J}_1^{N_{\rm dir}} \\ \vdots & & \vdots \\ \mathcal{J}_{N_{\rm dir}}^1 & \dots & \mathcal{J}_{N_{\rm dir}}^{N_{\rm dir}} \end{bmatrix}, \tag{3.9}$$

where the structure of each $2N_{\rm ant} \times 2N_{\rm ant}$ block at row c, column d, is exactly as given by Eq. 2.30 (or by Fig. 1f, in miniature).

The on-diagonal ("same-direction") blocks $\mathcal{J}_d^d$ will have the same structure as in the DI case (Eq. 2.12 or Eq. A3). Consider now the off-diagonal ("cross-direction") blocks $\mathcal{J}_c^d$. Their non-zero elements can take one of two forms:

$$\sum_{q\neq i,s} \bar{y}_{iqs}^{(c)} y_{iqs}^{(d)} = \sum_{q\neq i} g_q^{(c)} \bar{g}_q^{(d)} \sum_s \bar{m}_{iqs}^{(c)} m_{iqs}^{(d)} \tag{3.10}$$

or

$$\sum_s y_{jis}^{(c)} y_{ijs}^{(d)} = \bar{g}_i^{(c)} \bar{g}_j^{(d)} \sum_s \bar{m}_{ijs}^{(c)} m_{ijs}^{(d)} \tag{3.11}$$

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:

$$X_{pq}^{(cd)} = \left\langle \boldsymbol{m}_{pq}^{(\rm c)}, \boldsymbol{m}_{pq}^{(\rm d)} \right\rangle = \sum_s m_{pqs}^{(c)} \bar{m}_{pqs}^{(d)}. \tag{3.12}$$

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

$$m_{pqt\nu}^{(d)} = S_{pqt\nu}^{(d)} k_{pqt\nu}^{(d)}, \tag{3.13}$$

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

$$k_{pqt\nu}^{(d)} = e^{-2\pi i (\boldsymbol{u}_{pq}(t)\cdot\boldsymbol{\sigma}_d)\nu/c}, \quad \boldsymbol{\sigma}_d = [l_d, m_d, n_d - 1]^T, \tag{3.14}$$

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

$$X_{pq}^{(cd)} = S_{pq}^{(c)} S_{pq}^{(d)} \sum_s e^{-2\pi i [\boldsymbol{u}_{pq}(t)\cdot(\boldsymbol{\sigma}_c - \boldsymbol{\sigma}_d)]\nu/c} \tag{3.15}$$

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. 3.9 as null. The problem is then separable by direction, and $\boldsymbol{J}^H\boldsymbol{J}$ is approximated by a block-diagonal matrix:

$$\tilde{\boldsymbol{H}} = \begin{bmatrix} \mathcal{J}_1^1 & & 0 \\ & \ddots & \\ 0 & & \mathcal{J}_{N_{\rm dir}}^{N_{\rm dir}} \end{bmatrix}, \tag{3.16}$$

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.

3.2.2 COHJONES: separating by antenna

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

$$\boldsymbol{H}_{\rm U} = \begin{bmatrix} \boldsymbol{A}_1^1 & & 0 & \boldsymbol{B}_1^1 & \dots & \boldsymbol{B}_1^{N_{\rm ant}} \\ & \ddots & & \vdots & & \vdots \\ 0 & & \boldsymbol{A}_{N_{\rm ant}}^{N_{\rm ant}} & \boldsymbol{B}_{N_{\rm ant}}^1 & \dots & \boldsymbol{B}_{N_{\rm ant}}^{N_{\rm ant}} \end{bmatrix}, \tag{3.17}$$

that is, its left half is block-diagonal, consisting of $N_{\rm dir} \times N_{\rm dir}$ blocks (which follows from Eq. 2.30), while its right half consists of elements of the form given by Eq. 3.11.

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 $\tilde{\boldsymbol{H}}$ 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. 3.3 (noting that Eq. 2.20 holds):

$$\boldsymbol{g}_{k+1} = \tilde{\boldsymbol{H}}_{\rm UL}^{-1} \boldsymbol{J}_{\rm L}{}^H \check{\boldsymbol{d}}. \tag{3.18}$$

We may note in passing that for LM, the analogy is only approximate:

$$\breve{\boldsymbol{g}}_{k+1} \approx \frac{\lambda}{1+\lambda}\breve{\boldsymbol{g}}_k + \frac{1}{1+\lambda}\tilde{\boldsymbol{H}}_{\mathrm{UL}}^{-1}\boldsymbol{J}_{\mathrm{L}}{}^H\breve{\boldsymbol{d}}, \tag{3.19}$$

since $\tilde{\boldsymbol{H}}$ is only approximately diagonal.

This approach has been implemented as the CohJones (complex half-Jacobian optimization for n-directional estimation) algorithm[4], the results of which applied to simulated data are presented below.

3.2.3 ALLJONES: Separating all

Perhaps the biggest simplification available is to start with CDA or CAD ordering, and assume a StefCal-style diagonal approximation for the entirety of $\boldsymbol{J}^H\boldsymbol{J}$. The matrix then becomes purely diagonal, matrix inversion reduces to $O(N_{\mathrm{dir}}N_{\mathrm{ant}})$ in complexity, and algorithmic cost becomes dominated by the $O(N_{\mathrm{dir}}N_{\mathrm{ant}}^2)$ process of computing the diagonal elements of $\boldsymbol{J}^H\boldsymbol{J}$. Note that Eq. 2.20 no longer holds, and the GN update step must be computed via the residuals:

$$\boldsymbol{g}_{k+1} = \boldsymbol{g}_k + \tilde{\boldsymbol{H}}_{\mathrm{UL}}^{-1}\boldsymbol{J}_{\mathrm{L}}{}^H\breve{\boldsymbol{r}}, \tag{3.20}$$

with the per-element expression being

$$g_{p,k+1}^{(d)} = g_{p,k}^{(d)} + \Big(\sum_{q\neq p,s}\bar{y}_{pqs}^{(d)}y_{pqs}^{(d)}\Big)^{-1}\sum_{q\neq p,s}\bar{y}_{pqs}^{(d)}r_{pqs}. \tag{3.21}$$

3.3 Convergence and algorithmic cost

Evaluating the gain update (e.g. as given by Eq. 2.18) involves a number of computational steps:

(i) Computing $\tilde{\boldsymbol{H}} \approx \boldsymbol{J}^H\boldsymbol{J}$,
(ii) Inverting $\tilde{\boldsymbol{H}}$,
(iii) Computing the $\boldsymbol{J}^H\breve{\boldsymbol{d}}$ vector,
(iv) Multiplying the result of (ii) and (iii).

Since each of the algorithms discussed above uses a different sparse approximation for $\boldsymbol{J}^H\boldsymbol{J}$, each of these steps will scale differently (except iii, which is $O(N_{\mathrm{ant}}^2N_{\mathrm{dir}})$ for all algorithms). Table 2 summarizes the scaling behaviour. An additional scaling factor is given by the number of iterations required to converge. This is harder to quantify. For example, in our experience (Smirnov 2013), an "exact" (LM) implementation of DI calibration problem converges in much fewer iterations than StefCal (on the order of a few vs. a few tens), but is much slower in terms of "wall time" due to the more expensive iterations (N_{ant}^3 vs. N_{ant} scaling). This trade-off between "cheap–approximate" and "expensive–accurate" is typical for iterative algorithms.

CohJones accounts for interactions between directions, but ignores interactions between antennas. Early experience indicates that it converges in a few tens of iterations. AllJones uses the most approximative step of all, ignoring all interactions between parameters. Its convergence behaviour is untested at this time.

It is clear that depending on N_{ant} and N_{dir}, and also on the structure of the problem, there will be regimes where one or the other algorithm has a computational advantage. This should be investigated in a future work.

[4] In fact it was the initial development of CohJones by Tasse (2014) that directly led to the present work.

Table 2. The scaling of computational costs for a single iteration of the four DD calibration algorithms discussed in the text, broken down by computational step. The dominant term(s) in each case are marked by "†". Not shown is the cost of computing the $\boldsymbol{J}^H\breve{\boldsymbol{d}}$ vector, which is $O(N_{\mathrm{ant}}^2N_{\mathrm{dir}})$ for all algorithms. "Exact" refers to a naive implementation of GN or LM with exact inversion of the $\boldsymbol{J}^H\boldsymbol{J}$ term. Scaling laws for DI calibration algorithms may be obtained by assuming $N_{\mathrm{dir}} = 1$, in which case CohJones or AllJones become equivalent to StefCal.

algorithm	$\tilde{\boldsymbol{H}}$	$\tilde{\boldsymbol{H}}^{-1}$	multiply
Exact	$O(N_{\mathrm{ant}}^2N_{\mathrm{dir}}^2)$	$O(N_{\mathrm{ant}}^3N_{\mathrm{dir}}^3)^\dagger$	$O(N_{\mathrm{ant}}^2N_{\mathrm{dir}}^2)$
AllJones	$O(N_{\mathrm{ant}}^2N_{\mathrm{dir}})^\dagger$	$O(N_{\mathrm{ant}}N_{\mathrm{dir}})$	$O(N_{\mathrm{ant}}N_{\mathrm{dir}})$
CohJones	$O(N_{\mathrm{ant}}^2N_{\mathrm{dir}}^2)^\dagger$	$O(N_{\mathrm{ant}}N_{\mathrm{dir}}^3)^\dagger$	$O(N_{\mathrm{ant}}N_{\mathrm{dir}}^2)$
DCA	$O(N_{\mathrm{ant}}^2N_{\mathrm{dir}})$	$O(N_{\mathrm{ant}}^3N_{\mathrm{dir}})^\dagger$	$O(N_{\mathrm{ant}}^2N_{\mathrm{dir}})$

3.4 Smoothing in time and frequency

From physical considerations, we know that gains do not vary arbitrarily in 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. 2.3), 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 (Tasse 2014).

Another way to impose smoothness combines the ideas of solution intervals (Eq. 2.21) and weighting (Eq. 2.24). At every time/frequency sample t_0, ν_0, we can postulate a weighted LS problem:

$$\min_{\boldsymbol{g}}\sum_{pqt\nu} w(t-t_0,\nu-\nu_0)|r_{pqt\nu}|^2, \tag{3.22}$$

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 for a particularly elegant and efficient way of implementing this in practice. Consider the weighted equations of Eqs. 2.26 and 2.27, and replace the sample index s by t, ν. Under the diagonal approximation, each parameter update at t_0, ν_0 is computed as:

$$g_{p,k+1}(t_0,\nu_0) = \frac{\sum_{q\neq p,t,\nu} w(t-t_0,\nu-\nu_0)\bar{y}_{pqt\nu}d_{pqt\nu}}{\sum_{q\neq p,t,\nu} w(t-t_0,\nu-\nu_0)\bar{y}_{pqt\nu}y_{pqt\nu}}. \tag{3.23}$$

Looking at Eq. 3.23, it's clear that both sums represent a convolution. If we define two functions of t, ν:

$$\alpha_p(t,\nu) = \sum_{q\neq p}\bar{y}_{pqt\nu}d_{pqt\nu}, \quad \beta_p(t,\nu) = \sum_{q\neq p}\bar{y}_{pqt\nu}y_{pqt\nu}, \tag{3.24}$$

then Eq. 3.23 corresponds to the ratio of two convolutions

$$g_{p,k+1}(t,\nu) = \frac{w \circ \alpha_p}{w \circ \beta_p}, \tag{3.25}$$

sampled over a discrete t,ν grid. Note that the formulation above also allows for different smoothing kernels per antenna. Iterating Eq. 3.25 to convergence at every t,ν slot, we obtain per-antenna arrays of gain solutions answering Eq. 3.22. These solutions are smooth in frequency and time, with the degree of smoothness constrained by the kernel w.

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.

Smoothing may also be trivially incorporated into the AllJones algorithm, since its update step (Eq. 3.21) has exactly the same structure. A different smoothing kernel may be employed per direction (for example, directions further from the phase centre can employ a narrower kernel).

Since smoothing involves computing a $\boldsymbol{g}$ value at every t,ν point, rather than one value per solution interval, its computational costs are correspondingly higher. To put it another way, using solution intervals of size $N_t \times N_\nu$ introduces a savings of $N_t \times N_\nu$ (in terms of the number of invert and multiply steps required, see Table 2) over solving the problem at every t,ν slot; using smoothing foregoes these savings. In real implementations, this extra cost is mitigated by the fact that the computation given by Eq. 3.21 may be vectorized very efficiently over many t,ν slots. However, this vectorization is only straightforward because the matrix inversion in StefCal or AllJones reduces to simple scalar division. For CohJones or DCA this is no longer the case, so while smoothing may be incorporated into these algorithms in principle, it is not clear if this can be done efficiently in practice.

4 THE FULLY POLARIZED CASE

To incorporate polarization, let us start by rewriting the basic RIME of Eq. 2.1 using 2×2 matrices (a full derivation may be found in Smirnov 2011a):

$$\boldsymbol{D}_{pq} = \boldsymbol{G}_p \boldsymbol{M}_{pq} \boldsymbol{G}_q^H + \boldsymbol{N}_{pq}. \tag{4.1}$$

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

$$\min_{\{\boldsymbol{G}_p\}} \sum_{pq} ||\boldsymbol{R}_{pq}||_F, \quad \boldsymbol{R}_{pq} = \boldsymbol{D}_{pq} - \boldsymbol{G}_p \boldsymbol{M}_{pq} \boldsymbol{G}_q^H. \tag{4.2}$$

This is a set of 2×2 matrix equations, rather than the vector equations employed in the complex NNLS formalism above (Eq. 1.4). In principle, there is a straightforward way of recasting matrix equations into a form suitable to Eq. 1.4: we can vectorize each matrix equation, turning it into an equation on 4-vectors, and then derive the complex Jacobian in the usual manner (Eq. 1.7).

In this section we will obtain a more elegant derivation, by employing an operator calculus where the "atomic" elements are 2×2 matrices rather than scalars. This will allow us to define the Jacobian in a more transparent way, as a matrix of linear operators on 2×2 matrices. Mathematically, this is completely equivalent to vectorizing the problem and applying Wirtinger calculus (each matrix then corresponds to 4 elements of the parameter vector, and each operator in the Jacobian becomes a 4×4 matrix block), as will be shown in Appendix B.

4.1 Matrix operators and derivatives

By *matrix operator*, we shall refer to any function $\mathcal{F}$ that maps a 2×2 complex matrix to another such matrix:

$$\mathcal{F} : \mathbb{C}^{2\times2} \to \mathbb{C}^{2\times2}. \tag{4.3}$$

When the operator $\mathcal{F}$ is applied to matrix $\boldsymbol{X}$, we'll write the result as $\boldsymbol{Y} = \mathcal{F}\boldsymbol{X}$, or $\mathcal{F}[\boldsymbol{X}]$ if we need to avoid ambiguity.

If we fix a complex matrix $\boldsymbol{A}$, then two interesting (and linear) matrix operators are right-multiply by $\boldsymbol{A}$, and left-multiply by $\boldsymbol{A}$:

$$\begin{aligned} \mathcal{R}_{\boldsymbol{A}}\boldsymbol{X} &= \boldsymbol{X}\boldsymbol{A} \\ \mathcal{L}_{\boldsymbol{A}}\boldsymbol{X} &= \boldsymbol{A}\boldsymbol{X} \end{aligned} \tag{4.4}$$

Appendix B formally shows that all linear matrix operators, including $\mathcal{R}_{\boldsymbol{A}}$ and $\mathcal{L}_{\boldsymbol{A}}$, can be represented as multiplication of 4-vectors by 4×4 matrices.

Just from the operator definitions, it is trivial to see that

$$\mathcal{L}_{\boldsymbol{A}}\mathcal{L}_{\boldsymbol{B}} = \mathcal{L}_{\boldsymbol{AB}}, \quad \mathcal{R}_{\boldsymbol{A}}\mathcal{R}_{\boldsymbol{B}} = \mathcal{R}_{\boldsymbol{BA}}, \quad [\mathcal{R}_{\boldsymbol{A}}]^{-1} = \mathcal{R}_{\boldsymbol{A^{-1}}} \tag{4.5}$$

Consider now a matrix-valued function of n matrices and their Hermitian transposes

$$\boldsymbol{F}(\boldsymbol{G}_1 \ldots \boldsymbol{G}_n, \boldsymbol{G}_1^H \ldots \boldsymbol{G}_n^H), \tag{4.6}$$

and think what a meaningful definition for the partial matrix derivative $\partial\boldsymbol{F}/\partial\boldsymbol{G}_i$ would be. A derivative can be thought of as a local linear approximation to $\boldsymbol{F}$, i.e. a linear function mapping an increment in the argument $\Delta\boldsymbol{G}_i$ to an increment in the function value:

$$\boldsymbol{F}(...,\boldsymbol{G}_i + \Delta\boldsymbol{G}_i,...) - \boldsymbol{F}(...,\boldsymbol{G}_i,...) \approx \frac{\partial\boldsymbol{F}}{\partial\boldsymbol{G}_i}\Delta\boldsymbol{G}_i, \tag{4.7}$$

in other words, the partial derivative is a linear matrix operator, in the same sense that the Jacobian is a linear operator. Obviously, the linear operator that best approximates a linear operator is the operator itself, so we necessarily have

$$\frac{\partial(\boldsymbol{GA})}{\partial\boldsymbol{G}} = \mathcal{R}_A, \qquad \frac{\partial(\boldsymbol{AG^H})}{\partial\boldsymbol{G^H}} = \mathcal{L}_A. \tag{4.8}$$

Appendix B provides formal definitions of the *Wirtinger matrix derivatives*

$$\frac{\partial\boldsymbol{F}}{\partial\boldsymbol{G}}, \quad \frac{\partial\boldsymbol{F}}{\partial\boldsymbol{G}^H} \tag{4.9}$$

that are completely equivalent to the partial complex Jacobians defined earlier.

Note that this calculus also offers a natural way of taking more complicated matrix derivatives (that is, for more elaborate versions of the RIME). For example,

$$\frac{\partial(\boldsymbol{AGB})}{\partial \boldsymbol{G}} = \mathcal{L}_A \mathcal{R}_B, \tag{4.10}$$

which is a straightforward manifestation of the chain rule: $\boldsymbol{AGB} = \boldsymbol{A}(\boldsymbol{GB})$.

4.2 Complex Jacobians for the polarized RIME

Let us now apply this operator calculus to Eq. 4.2. Taking the derivatives, we have:

$$\frac{\partial \boldsymbol{V}_{pq}}{\partial \boldsymbol{G}_p} = \mathcal{R}_{\boldsymbol{M}_{pq}\boldsymbol{G}_q^H}, \quad \text{and} \quad \frac{\partial \boldsymbol{V}_{pq}}{\partial \boldsymbol{G}_q^H} = \mathcal{L}_{\boldsymbol{G}_p\boldsymbol{M}_{pq}}. \tag{4.11}$$

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

$$\breve{\mathbf{G}} = [\boldsymbol{G}_1, \ldots, \boldsymbol{G}_{N_{\text{ant}}}, \boldsymbol{G}_1^H, \ldots, \boldsymbol{G}_{N_{\text{ant}}}^H]^T, \tag{4.12}$$

then we may construct the top half of the full complex Jacobian operator in full analogy with the derivation of Eq. 2.6. 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 $p < q$).

$$\boldsymbol{J}_{\text{U}} = \Big[\overbrace{\mathcal{R}_{\boldsymbol{M}_{pq}\boldsymbol{G}_q^H}\delta_p^j}^{j=1\ldots N_{\text{ant}}} \quad \overbrace{\mathcal{L}_{\boldsymbol{G}_p\boldsymbol{M}_{pq}}\delta_q^j}^{j=1\ldots N_{\text{ant}}} \Big] \Big\}_{[pq]=1\ldots N_{\text{bl}}\ (p<q)} \tag{4.13}$$

Note that there are two fully equivalent ways to read the above equation. In operator notation, it specifies a linear operator $\boldsymbol{J}_{\text{U}}$ that maps a $2N_{\text{ant}}$-vector of 2×2 matrices to an N_{bl}-vector of 2×2 matrices. In conventional matrix notation (Appendix B), $\boldsymbol{J}_{\text{U}}$ 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. 1.7, this corresponds to the derivatives of the conjugate residual vector $\bar{\boldsymbol{r}}_k$, and can be constructed by conjugating and mirroring $\boldsymbol{J}_{\text{U}}$. Let us modify this construction by taking the derivative of the Hermitian transpose of the residuals instead. Note that substituting the Hermitian transpose for element-by-element conjugation corresponds to a simple reordering of some rows in the conjugate residual vector, which we always are free to do. Let us then construct the augmented residual vector of matrices as:

$$\breve{\mathbf{R}} = \begin{bmatrix} \boldsymbol{R}_{pq} \\ \boldsymbol{R}_{pq}^H \end{bmatrix} \begin{matrix} \}_{[pq]=1\ldots N_{\text{bl}}\ (p<q)} \\ \}_{[pq]=1\ldots N_{\text{bl}}\ (p<q)} \end{matrix} \tag{4.14}$$

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

$$\frac{\partial \boldsymbol{V}_{pq}^H}{\partial \boldsymbol{G}_q} = \mathcal{R}_{\boldsymbol{M}_{pq}^H\boldsymbol{G}_p^H}, \quad \text{and} \quad \frac{\partial \boldsymbol{V}_{pq}^H}{\partial \boldsymbol{G}_p^H} = \mathcal{L}_{\boldsymbol{G}_q\boldsymbol{M}_{pq}^H}, \tag{4.15}$$

and we may write out the full complex Jacobian as

$$\boldsymbol{J} = -\begin{bmatrix} \mathcal{R}_{\boldsymbol{M}_{pq}\boldsymbol{G}_q^H}\delta_p^j & \mathcal{L}_{\boldsymbol{G}_p\boldsymbol{M}_{pq}}\delta_q^j \\ \mathcal{R}_{\boldsymbol{M}_{pq}^H\boldsymbol{G}_p^H}\delta_q^j & \mathcal{L}_{\boldsymbol{G}_q\boldsymbol{M}_{pq}^H}\delta_p^j \end{bmatrix} \begin{matrix} \}_{[pq]=1\ldots N_{\text{bl}}\ (p<q)} \\ \}_{[pq]=1\ldots N_{\text{bl}}\ (p<q)} \end{matrix} \tag{4.16}$$

We may now make exactly the same observation as we did to derive Eq. 2.8, and rewrite both $\boldsymbol{J}$ and $\breve{\mathbf{R}}$ in terms of a single row block. The pq index will now run through $2N_{\text{bl}}$ values (i.e. enumerating all combinations of $p \neq q$):

$$\boldsymbol{J} = \begin{bmatrix} \mathcal{R}_{\boldsymbol{M}_{pq}\boldsymbol{G}_q^H}\delta_p^j & \mathcal{L}_{\boldsymbol{G}_p\boldsymbol{M}_{pq}}\delta_q^j \end{bmatrix} \Big\}_{[pq]=1\ldots 2N_{\text{bl}}\ (p\neq q)} \tag{4.17}$$

and

$$\breve{\mathbf{R}} = \begin{bmatrix} \boldsymbol{R}_{pq} \end{bmatrix} \Big\}_{[pq]=1\ldots 2N_{\text{bl}}\ (p\neq q)} \tag{4.18}$$

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

$$\boldsymbol{Y}_{pq} = \boldsymbol{M}_{pq}\boldsymbol{G}_q^H, \tag{4.19}$$

and, noting that

$$\boldsymbol{Y}_{qp}^H = \boldsymbol{G}_p\boldsymbol{M}_{pq}, \tag{4.20}$$

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

$$\boldsymbol{J}^H\boldsymbol{J} = \begin{bmatrix} \operatorname{diag} \sum_{q\neq i} \mathcal{R}_{\boldsymbol{Y}_{iq}\boldsymbol{Y}_{iq}^H} & \nearrow^H \\ \hdashline \begin{cases} \mathcal{L}_{Y_{ji}}\mathcal{R}_{\boldsymbol{Y}_{ij}}, & i\neq j \\ 0, & i=j \end{cases} & \searrow \end{bmatrix} \tag{4.21}$$

(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. 2.12.

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

$$\boldsymbol{J}^H\breve{\mathbf{R}} = \begin{bmatrix} \sum_{pq} \boldsymbol{Y}_{pq}^H\boldsymbol{R}_{pq}\delta_p^i \\ \hdashline \sum_{pq} \boldsymbol{R}_{pq}\boldsymbol{Y}_{qp}\delta_q^i \end{bmatrix} = \begin{bmatrix} \sum_{q\neq i} \boldsymbol{Y}_{iq}^H\boldsymbol{R}_{iq} \\ \hdashline \downarrow^H \end{bmatrix}, \tag{4.22}$$

where the second equality is established by swapping the p and q indices. Unsurprisingly, and by analogy with Eq. 2.14, the bottom half of the vector is Hermitian with respect to the top.

4.3 Parameter updates and the diagonal approximation

The relation of Eq. 2.16 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

$$\breve{\mathbf{D}} = [\boldsymbol{D}_{pq}], \quad \breve{\mathbf{V}} = [\boldsymbol{G}_p\boldsymbol{M}_{pq}\boldsymbol{G}_q^H], \tag{4.23}$$

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

$$\breve{\mathbf{G}}_{k+1} = (\boldsymbol{J}^H\boldsymbol{J})^{-1}\boldsymbol{J}^H\breve{\mathbf{D}}. \tag{4.24}$$

By analogy with Eq. 2.18, this equation also holds when $\boldsymbol{J}^H\boldsymbol{J}$ is approximated, but only if the condition of Eq. 2.20 is met.

To actually implement GN or LM optimization, we still need to invert the operator form of the $\boldsymbol{J}^H\boldsymbol{J}$ matrix in Eq. 4.21. We have two options here. The brute-force numerical approach is to substitute the 4×4 matrix forms of the $\mathcal{R}$ and $\mathcal{L}$ operators (Eqs. B7 and B8) 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. 3.1. If we neglect the off-diagonal operators of Eq. 4.21, 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 can therefore directly invert the operator form of $\boldsymbol{J}^H\boldsymbol{J}$ and apply it to $\boldsymbol{J}^H\breve{\mathbf{R}}$, thus arriving at a simple per-antenna equation for the GN update step:

$$\boldsymbol{G}_{p,k+1} = \left[\sum_{q\neq p} \boldsymbol{Y}^H_{pq}\boldsymbol{D}_{pq}\right]\left[\sum_{q\neq p}\boldsymbol{Y}_{pq}\boldsymbol{Y}^H_{pq}\right]^{-1} \tag{4.25}$$

This is, once again, equivalent to the polarized StefCal update step proposed by Smirnov (2013) and Salvini & Wijnholds (2014a).

4.4 Polarized direction-dependent calibration

Let us now briefly address the fully-polarized DD case. This can be done by direct analogy with Sect. 2.5, using the operator calculus developed above. As a result, we arrive at the following expression for $\boldsymbol{J}^H\boldsymbol{J}$:

$$\boldsymbol{J}^H\boldsymbol{J} = \begin{bmatrix} \delta^i_j \sum\limits_{q\neq i,s} \mathcal{R}_{\boldsymbol{Y}^{(d)}_{iqs}\boldsymbol{Y}^{(c)H}_{iqs}} & \big| \nearrow^H \\ \hdashline \begin{cases} \sum\limits_s \mathcal{L}_{\boldsymbol{Y}^{(c)}_{jis}}\mathcal{R}_{\boldsymbol{Y}^{(d)}_{ijs}}, & i\neq j \\ 0 & i=j \end{cases} & \big| \searrow \end{bmatrix}, \tag{4.26}$$

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

$$\boldsymbol{J}^H\breve{\mathbf{R}} = \begin{bmatrix} \sum\limits_{q\neq i,s} \boldsymbol{Y}^{(d)H}_{iqs}\boldsymbol{R}_{iqs} \\ \hdashline \downarrow^H \end{bmatrix}. \tag{4.27}$$

All the separability considerations of Sect. 3 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 $\tilde{\boldsymbol{H}}$ matrix is fully diagonal in operator form, and the GN update step can be computed as

$$\delta\boldsymbol{G}^{(d)}_{p,k+1} = \left[\sum_{q\neq p,s}\boldsymbol{Y}^{(d)H}_{pqs}\boldsymbol{R}_{pqs}\right]\left[\sum_{q\neq p,s}\boldsymbol{Y}^{(d)}_{pqs}\boldsymbol{Y}^{(d)H}_{pqs}\right]^{-1}. \tag{4.28}$$

Note that in this case (as in the unpolarized AllJones version) the condition of Eq. 2.20 is not met, so we must use the residuals and compute $\delta\boldsymbol{G}$.

- If we only assume separability by antenna, as in the CohJones algorithm, then the $\tilde{\boldsymbol{H}}$ matrix becomes $4N_{\rm dir}\times 4N_{\rm dir}$-block-diagonal, and may be inverted exactly at a cost of $O(N^3_{\rm dir}N_{\rm ant})$. The condition of Eq. 2.20 is met.

It is also straightfoward to add weights and/or sliding window averging to this formulation, as per Sect. 2.4 and 3.4.

Equations 4.26–4.27 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. 4.28 and its ilk are 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.

5 OTHER DD ALGORITHMIC VARIATIONS

The mathematical framework developed above (in particular, Eqs. 4.26–4.27) 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. 3 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 (2014a). 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.

5.1 Feed forward

Salvini & Wijnholds (2014b) propose variants of the StefCal algorithm ("2-basic" and "2-relax") where the results of the update step (Eq. 4.25, 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 $\boldsymbol{G}_k$ (via the appropriate $\boldsymbol{Y}$ terms). This is shown to substantially improve convergence, at the cost of sacrificing the embarassing parallellism 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. 4.28) could implement feed-forward by both antenna (via $\boldsymbol{Y}$) and by direction – by recomputing the residuals $\boldsymbol{R}$ to take into account the updated solutions for $\boldsymbol{G}^{(1)}\ldots\boldsymbol{G}^{(d-1)}$ before evaluating the solution for $\boldsymbol{G}^{(d)}$. 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.

5.2 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:

$$\tilde{\boldsymbol{H}} = \begin{bmatrix} \mathcal{J}^1_1 & 0 & \cdots & 0 \\ \mathcal{J}^1_2 & \mathcal{J}^2_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ \mathcal{J}^1_N & \mathcal{J}^2_N & \cdots & \mathcal{J}^N_N \end{bmatrix}, \tag{5.1}$$

The inverse of this is also block triangular:

$$\tilde{\boldsymbol{H}}^{-1} = \begin{bmatrix} \mathcal{K}^1_1 & 0 & \cdots & 0 \\ \mathcal{K}^1_2 & \mathcal{K}^2_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ \mathcal{K}^1_N & \mathcal{K}^2_N & \cdots & \mathcal{K}^N_N \end{bmatrix}, \tag{5.2}$$

which can be computed using Gaussian elimination:

$$\begin{aligned}
\mathcal{K}_1^1 &= [\mathcal{J}_1^1]^{-1} \\
\mathcal{K}_2^2 &= [\mathcal{J}_2^2]^{-1} \\
\mathcal{K}_2^1 &= -\mathcal{K}_2^2 \mathcal{J}_2^1 \mathcal{K}_1^1 \\
\mathcal{K}_3^3 &= [\mathcal{J}_3^3]^{-1} \\
\mathcal{K}_3^2 &= -\mathcal{K}_3^3 \mathcal{J}_3^2 \mathcal{K}_2^2 \\
\mathcal{K}_3^1 &= -\mathcal{K}_3^3 [\mathcal{J}_3^1 \mathcal{K}_1^1 + \mathcal{J}_3^2 \mathcal{K}_2^1] \\
&\dots
\end{aligned} \tag{5.3}$$

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

5.3 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.

The term "peeling" comes from step (iv), since sources are "peeled" away one at a time[5].

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, since a second cycle tends to improve the solutions.

5.4 Exact matrix inversion

Better approximations to $(\boldsymbol{J}^H\boldsymbol{J})^{-1}$ (or a faster exact inverse) may exist. Consider, for example, Fig. 1f: 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.

[5] 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.

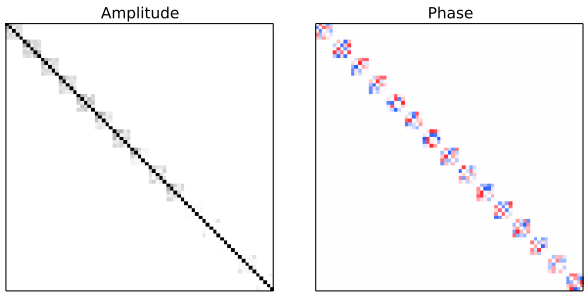

Figure 3. Amplitude (left panel) and phase (right panel) of the block-diagonal matrix $(\boldsymbol{J}^H\boldsymbol{J})_{\rm UL}$ for the dataset described in the text. Each block corresponds to one antenna; the pixels within a block correspond to directions.

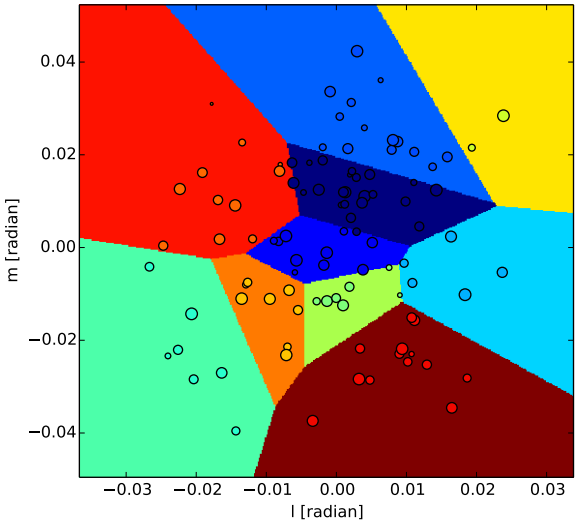

Figure 5. In order to conduct direction-dependent calibration, sources are clustered using a Voronoi tessellation algorithm. Each cluster has its own DD gain solution.

6 IMPLEMENTATIONS

6.1 StefCal in MeqTrees

Some of the ideas above have already been implemented in the MeqTrees (Noordam & Smirnov 2010) version of StefCal (Smirnov 2013). In particular, the MeqTrees version uses peeling (Sect. 5.3) to deal with DD solutions, and implements fully polarized StefCal with support for both solution intervals and time/frequency smoothing with a Gaussian kernel (as per Sect. 3.4). This has already been applied to JVLA L-band data to obtain what is (at time of writing) a world record dynamic range (3.2 million) image of the field around 3C147 (Perley 2013).

6.2 CohJones tests with simulated data

The CohJones algorithm, in the unpolarized version, has been implemented as a standalone Python script that uses

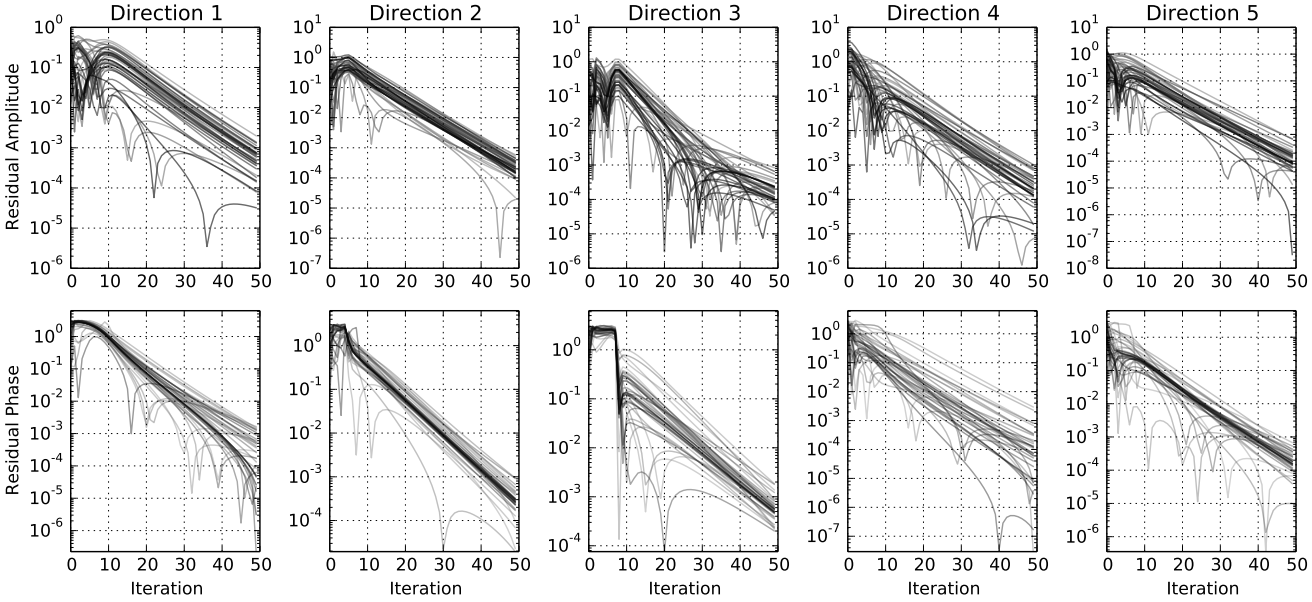

Figure 2. Amplitude (top row) and phase (bottom row) of the difference between the estimated and true gains, as a function of iteration. Columns correspond to directions. Different lines correspond to different antennas.

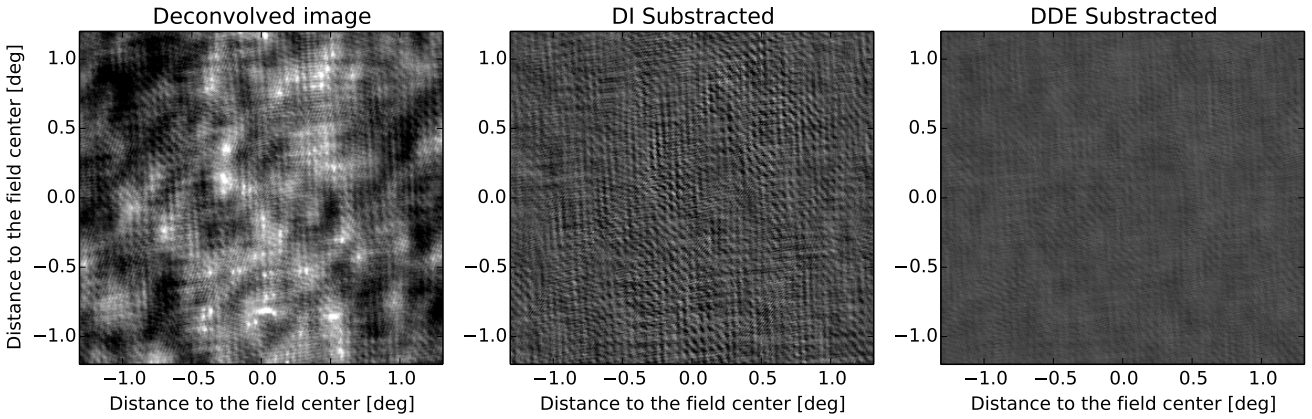

Figure 4. Simulation with time-variable DD gains. We show a deconvolved image (left) where no DD solutions have been applied, a residual image (centre) made by subtracting the sky model (in the visibility plane) without any DD corrections, and a residual image (right) made by subtracting the sky model with CohJones-estimated DD gain solutions (right). The color scale is the same in all panels. In this simulation, applying CohJones for DD calibration reduces the residual rms level by a factor of ~ 4.

the `pyrap`[6] and `casacore`[7] libraries to interface to Measurement Sets. This section reports on tests of our implementation with simulated Low Frequency Array (LOFAR) data.

For the tests, we build a dataset using a LOFAR layout with 40 antennas. The phase center is located at $\delta = +52^\circ$, the observing frequency is set to 50 MHz (single channel), and the integrations are 10s. We simulate 20 minutes of data.

For the first test, we use constant direction-dependent gains. We then run CohJones with a single solution interval corresponding to the entire 20 minutes. This scenario is essentially just a test of convergence. For the second test, we simulate a physically realistic time-variable ionosphere to derive the simulated DD gains.

6.2.1 Constant DD gains

To generate the visibilities for this test, we use a sky model containing five sources in an "+" shape, separated by 1°. The gains for each antenna p, direction d are constant in time, and are taken at random along a normal distribution $g_p^{(d)} \sim \mathcal{N}(0,1) + i\mathcal{N}(0,1)$. The data vector $\breve{\boldsymbol{d}}$ is then built from all baselines, and the full 20 minutes of data. The solution interval is set to the full 20 minutes, so a single solution per direction, per antenna is obtained.

The corresponding matrix $(\boldsymbol{J}^H\boldsymbol{J})_{\rm UL}$ is shown in Fig. 3.

[6] `https://code.google.com/p/pyrap`
[7] `https://code.google.com/p/casacore`

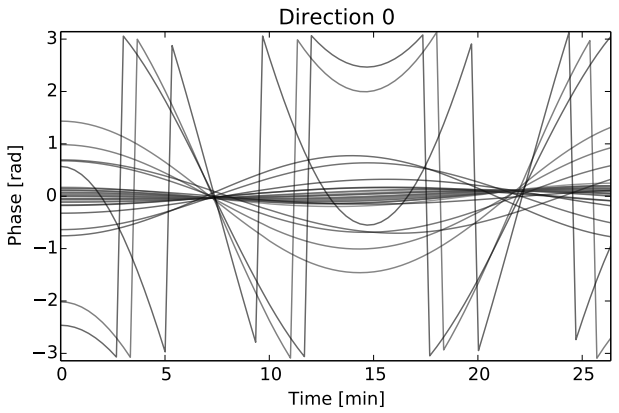

Figure 6. The complex phases of the DD gain terms (for all antennas and a single direction) derived from the time-variable TEC screen used in Sect. 6.2.2.

It is block diagonal, each block having size $N_{\rm dir} \times N_{\rm dir}$. The convergence of gain solutions as a function of direction is shown in Fig. 2. It is important to note that the problem becomes better conditioned (and CohJones converges faster) as the blocks of $(\boldsymbol{J}^H\boldsymbol{J})_{\rm UL}$ become more diagonally-dominated (equivalently, as the sky model components become more orthogonal). As discussed in Sect. 3.2.1, this happens (i) when more visibilities are taken into account (larger solution intervals) or (ii) if the directions are further away from each other.

6.2.2 Time-variable DD gains

To simulate a more realistic dataset, we use a sky model composed of 100 point sources of random (uniformly distributed) flux density. We also add noise to the visibilities, at a level of about 1% of the total flux. We simulate (scalar, phase-only) DD gains, using an ionospheric model consisting of a simple phase screen (an infinitesimally thin layer at a height of 100 km). The total electron content (TEC) values at the set of sample points are generated using Karhunen-Loeve decomposition (the spatial correlation is given by Kolmogorov turbulence, see van der Tol 2009). The constructed TEC-screen has an amplitude of ~ 0.07 TEC-Unit, and the corresponding DD phase terms are plotted in Fig. 6.

For calibration purposes, the sources are clustered in 10 directions using Voronoi tesselation (Fig. 5). The solution time-interval is set to 4 minutes, and a separate gain solution is obtained per each direction. Fig. 4 shows images generated from the residual visibilities, where the best-fitting model is subtracted in the visibility domain. The rms residuals after CohJones has been applied are a factor of ~ 4 lower than without DD solutions.

CONCLUSIONS

Recent developments in optimization theory have extended traditional NLLS optimization approaches to functions of complex variables. We have applied this to radio interferometric gain calibration, and shown that the use of complex Jacobians allow for new insights into the problem, leading to the formulation of a whole new family of DI and DD calibration algorithms. These algorithms hinge around different sparse approximations of the $\boldsymbol{J}^H\boldsymbol{J}$ matrix; we show that some recent algorithmic developments, notably StefCal, naturally fit into this framework as a particular special case of sparse (specifically, diagonal) approximation.

The proposed algorithms have different scaling properties depending on the selected matrix approximation – in all cases better than the cubic scaling of brute-force GN or LM methods – and may therefore exhibit different computational advantages depending on the dimensionality of the problem (number of antennas, number of directions). We also demonstrate an implementation of one particular algorithm for DD gain calibration, CohJones.

The use of complex Jacobians results in relatively compact and simple equations, and the resulting algorithms tend to be embarassingly parallel, which makes them particularly amenable to implementation on new massively-parallel computing architectures such as GPUs.

Complex optimization is applicable to a broader range of problems. Solving for a large number of independent DD gain parameters is not always desirable, as it potentially makes the problem under-constrained, and can lead to artefacts such as ghosts and source suppression. The alternative is solving for DD effect models that employ [a smaller set of] physical parameters, such as parameters of the primary beam and ionosphere. If these parameters are complex, then the complex Jacobian approach applies. Finally, although this paper only treats the NLLS problem (thus implicitly assuming Gaussian statistics), the approach is valid for the general optimization problem as well.

Other approximations or fast ways of inverting the complex $\boldsymbol{J}^H\boldsymbol{J}$ matrix may exist, and future work can potentially yield new and faster algorithms within the same unifying mathematical framework. This flexibility is particularly important for addressing the computational needs of the new generation of the so-called "SKA pathfinder" telescopes, as well as the SKA itself.

ACKNOWLEDGMENTS

We would like to thank the referee, Johan Hamaker, for extremely valuable comments that improved the paper. This work is based upon research supported by the South African Research Chairs Initiative of the Department of Science and Technology and National Research Foundation. Trienko Grobler originally pointed us in the direction of Wirtinger derivatives.

REFERENCES

Hamaker J. P., 2000, A&AS, 143, 515
Hamaker J. P., Bregman J. D., Sault R. J., 1996, A&AS, 117, 137
Kazemi S., Yatawatta S., 2013, MNRAS, 435, 597
Kreutz-Delgado K., 2009, arXiv:math/0906.4835
Laurent S., van Barel M., de Lathauwer L., 2012, SIAM J. Optim., 22, 879
Madsen K., Nielsen H. B., Tingleff O., 2004, Methods For Non-linear Least Squares Problems. Informatics and

Mathematical Modelling, Technical University of Denmark
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., Groundbased Telescopes Vol. 5489 of Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, LOFAR calibration challenges. p. 817
Noordam J. E., Smirnov O. M., 2010, A&A, 524, A61
Perley R., 2013, High Dynamic Range Imaging, presentation at "The Radio Universe @ Ger's (wave)-length" conference (Groningen, November 2013), http://www.astron.nl/gerfeest/presentations/perley.pdf
Salvini S., Wijnholds S. J., 2014a, A&A, 571, A97
Salvini S., Wijnholds S. J., 2014b, in General Assembly and Scientific Symposium (URSI GASS), 2014 XXXIth URSI Stefcal – an alternating direction implicit method for fast full polarization array calibration. pp 1–4
Smirnov O. M., 2011a, A&A, 527, A106
Smirnov O. M., 2011b, A&A, 527, A107
Smirnov O. M., 2011c, A&A, 527, A108
Smirnov O. M., 2013, StefCal: The fastest selfcal in the West, presentation at 3GC3 workshop (Port Alfred, February 2013), http://tinyurl.com/pzu8hco
Tasse C., 2014, arXiv:astro-ph/1410.8706
Tasse C., 2014, A&A, 566, A127
van der Tol S., 2009, PhD thesis, TU Delft, pp 1185–1205
Wirtinger W., 1927, Mathematische Annalen, 97, 357

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. 2.4 and 2.5):

$$\boldsymbol{J}_k = \begin{bmatrix} m_{12}\bar{g}_2 & 0 & 0 \\ m_{13}\bar{g}_3 & 0 & 0 \\ 0 & m_{23}\bar{g}_3 & 0 \end{bmatrix}, \boldsymbol{J}_{k^*} = \begin{bmatrix} 0 & g_1 m_{12} & 0 \\ 0 & 0 & g_1 m_{13} \\ 0 & 0 & g_2 m_{23} \end{bmatrix} \tag{A1}$$

We then get the following expression for the full complex Jacobian $\boldsymbol{J}$ (Eq. 2.8):

$$\begin{bmatrix} m_{12}\bar{g}_2 & 0 & 0 & 0 & g_1 m_{12} & 0 \\ m_{13}\bar{g}_3 & 0 & 0 & 0 & 0 & g_1 m_{13} \\ 0 & m_{21}\bar{g}_1 & 0 & g_2 m_{21} & 0 & 0 \\ 0 & m_{23}\bar{g}_3 & 0 & 0 & 0 & g_2 m_{23} \\ 0 & 0 & m_{31}\bar{g}_1 & g_1 m_{31} & 0 & 0 \\ 0 & 0 & m_{32}\bar{g}_2 & 0 & g_3 m_{32} & 0 \end{bmatrix} \tag{A2}$$

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

$$\begin{bmatrix} y_{12}^2+y_{13}^2 & 0 & 0 & 0 & \bar{y}_{12}\bar{y}_{21} & \bar{y}_{13}\bar{y}_{31} \\ 0 & y_{12}^2+y_{23}^2 & 0 & \bar{y}_{12}\bar{y}_{21} & 0 & \bar{y}_{23}\bar{y}_{32} \\ 0 & 0 & y_{13}^2+y_{23}^2 & \bar{y}_{13}\bar{y}_{31} & \bar{y}_{23}\bar{y}_{32} & 0 \\ 0 & y_{12}y_{21} & y_{13}y_{31} & y_{12}^2+y_{13}^2 & 0 & 0 \\ y_{12}y_{21} & 0 & y_{23}y_{32} & 0 & y_{12}^2+y_{23}^2 & 0 \\ y_{13}y_{31} & y_{23}y_{32} & 0 & 0 & 0 & y_{13}^2+y_{23}^2 \end{bmatrix} \tag{A3}$$

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

$$\boldsymbol{J}^H\check{\boldsymbol{r}} = \begin{bmatrix} \bar{y}_{12}r_{12} + \bar{y}_{13}r_{13} \\ \bar{y}_{21}r_{21} + \bar{y}_{23}r_{23} \\ \bar{y}_{31}r_{31} + \bar{y}_{32}r_{32} \\ y_{12}\bar{r}_{12} + y_{13}\bar{r}_{13} \\ y_{21}\bar{r}_{21} + y_{23}\bar{r}_{23} \\ y_{31}\bar{r}_{31} + y_{32}\bar{r}_{32} \end{bmatrix}. \tag{A4}$$

APPENDIX B: OPERATOR CALCULUS

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

$$\operatorname{vec} \boldsymbol{X} = \begin{bmatrix} x_{11} \\ x_{21} \\ x_{12} \\ x_{22} \end{bmatrix}, \quad \operatorname{vec}^{-1} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{12} \\ x_{22} \end{bmatrix} = \boldsymbol{X}, \tag{B1}$$

which sets up a trivial isomorphism between the space of 2×2 complex matrices $\mathbb{C}^{2\times 2}$ and the space $\mathbb{C}^4$. Note that the "vec" operator is linear. Therefore, any linear operator on $\boldsymbol{X} \in \mathbb{C}^{2\times 2}$, which we'll write as $\mathcal{B}\boldsymbol{X}$, must be equivalent to multiplication of $\boldsymbol{x} = \operatorname{vec} \boldsymbol{X} \in \mathbb{C}^4$ by some 4×4 complex matrix $\boldsymbol{B}$, and vice versa:

$$\mathbf{x} = \operatorname{vec} \boldsymbol{X}, \quad \boldsymbol{B}\mathbf{x} = \operatorname{vec} \mathcal{B}\boldsymbol{X} \tag{B2}$$

To put this formally, we can say that the vec operator induces an isomorphism $\mathcal{W}$ between the space of linear operators on $\mathbb{C}^4$ – that is, $\mathbb{C}^{4\times 4}$, and the space of linear operators on $\mathbb{C}^{2\times 2}$:

$$\begin{aligned} &\mathbb{C}^{4\times 4} = \operatorname{Lin}(\mathbb{C}^4, \mathbb{C}^4) \xrightarrow{\mathcal{W}} \operatorname{Lin}(\mathbb{C}^{2\times 2}, \mathbb{C}^{2\times 2}) \\ &\mathcal{W}(\boldsymbol{B})[\boldsymbol{X}] = \operatorname{vec}^{-1}(\boldsymbol{B} \operatorname{vec} \boldsymbol{X}) \\ &\mathcal{W}^{-1}(\mathcal{B})\boldsymbol{x} = \operatorname{vec}(\mathcal{B}[\operatorname{vec}^{-1} \boldsymbol{x}]) \end{aligned} \tag{B3}$$

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

$$\mathcal{R}_{\boldsymbol{A}}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{A}, \quad \mathcal{L}_{\boldsymbol{A}}\boldsymbol{X} = \boldsymbol{A}\boldsymbol{X} \tag{B4}$$

The $\mathcal{W}$ isomorphism ensures that these operators can be represented as multiplication of 4-vectors by specific kinds of 4×4 matrices. The matrix outer product proves to be useful here, and in particular the following basic relation:

$$\operatorname{vec}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}) = (\boldsymbol{C}^T \otimes \boldsymbol{A}) \operatorname{vec} \boldsymbol{B}, \tag{B5}$$

from which we can derive the matrix operator forms via:

$$\begin{aligned} \operatorname{vec}(\mathbb{I}\boldsymbol{X}\boldsymbol{A}) &= (\boldsymbol{A}^T \otimes \mathbb{I}) \operatorname{vec} \boldsymbol{X} \\ \operatorname{vec}(\boldsymbol{A}\boldsymbol{X}\mathbb{I}) &= (\mathbb{I} \otimes \boldsymbol{A}) \operatorname{vec} \boldsymbol{X}, \end{aligned} \tag{B6}$$

which gives us

$$\operatorname{vec}\mathcal{R}_{\boldsymbol{A}}[\boldsymbol{X}] = \begin{bmatrix} a_{11} & 0 & a_{21} & 0 \\ 0 & a_{11} & 0 & a_{21} \\ a_{12} & 0 & a_{22} & 0 \\ 0 & a_{12} & 0 & a_{22} \end{bmatrix} \operatorname{vec}\boldsymbol{X} \tag{B7}$$

and

$$\operatorname{vec}\mathcal{L}_{\boldsymbol{A}}[\boldsymbol{X}] = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ 0 & 0 & a_{11} & a_{12} \\ 0 & 0 & a_{21} & a_{22} \end{bmatrix} \operatorname{vec}\boldsymbol{X}. \tag{B8}$$

Note that Eq. 4.5, which we earlier derived from the operator definitions, can now be verified with the 4×4 forms. Note also that Eq. 4.5 is equally valid whether interpreted in terms of chaining operators, or multipying the equivalent 4×4 matrices.

B1 Derivative operators and Jacobians

Consider a matrix-valued function of a a matrix argument and its Hermitian transpose, $\boldsymbol{F}(\boldsymbol{G}, \boldsymbol{G}^H)$. The "vec" operator induces a natural isomorphism between such functions and 4-vector valued functions of 4-vectors:

$$\boldsymbol{f}(\boldsymbol{g}, \bar{\boldsymbol{g}}) = \operatorname{vec}\boldsymbol{F}(\operatorname{vec}^{-1}\boldsymbol{g}, (\operatorname{vec}^{-1}\bar{\boldsymbol{g}})^T). \tag{B9}$$

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 given by Eq. 2.4:

$$\boldsymbol{J}_k = [\partial f_i/\partial g_j], \quad \boldsymbol{J}_{k^*} = [\partial f_i/\partial \bar{g}_j], \tag{B10}$$

that represent linear approximations to $\boldsymbol{f}$, i.e. linear operators on $\mathbb{C}^4$ mapping increments in the arguments $\Delta\boldsymbol{g}$ and $\Delta\bar{\boldsymbol{g}}$ to increments in the function value $\Delta\boldsymbol{f}$. The isomorphism defined above maps these to the equivalent linear operators on $\mathbb{C}^{2\times2}$ that represent linear approximations to $\boldsymbol{F}$. It is the latter operators that we call *the Wirtinger matrix derivatives* of $\boldsymbol{F}$ with respect to $\boldsymbol{G}$ and $\boldsymbol{G}^{\boldsymbol{H}}$:

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{G}} = \mathcal{W}(\boldsymbol{J}_k), \quad \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{G}^H} = \mathcal{W}(\boldsymbol{J}_{k^*}^T) \tag{B11}$$

This is more than just a formal definition: thanks to the isomorphism, the operators given by Eq. B11 are Wirtinger derivatives in exactly the same sense that the Jacobians of Eq. B10 are Wirtinger derivatives, with the former being simply the $\mathbb{C}^{2\times2}$ manifestation of the gradient operators in $\mathbb{C}^4$, as defined in Sect. 1. However, operating in $\mathbb{C}^{2\times2}$ space allows us to write the larger Jacobians of Sect. 4 in terms of simpler matrices composed of operators, resulting in a Jacobian structure that is entirely analogous to the scalar derivation.

APPENDIX C: GRADIENT-BASED OPTIMIZATION ALGORITHMS

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

C1 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):

$$\delta\check{\boldsymbol{z}}_k = -\lambda \boldsymbol{J}^H \check{\boldsymbol{r}}_k, \tag{C1}$$

where λ is some small value;
(iv) If not converged[8], set $\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \delta\boldsymbol{z}$, and go back to step (ii).

C2 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. 1.9 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).

C3 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. 1.9 (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).

C4 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 predefined threshold: $\chi^2_{k-1} - \chi^2_k < \epsilon_0$.
- Norm of the gradient smaller than some threshold: $||\boldsymbol{J}||_F < \gamma_0$.

[8] see below