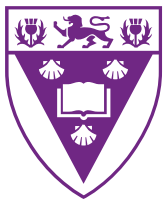

RHODES UNIVERSITY

Where leaders learn

CubiCal: A fast radio interferometric calibration suite exploiting complex optimisation

Author:
Jonathan KENYON

Supervisor:
Prof Oleg SMIRNOV

Co-supervisor:
Dr. Trienko GROBLER

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

of

Rhodes University

March, 2019

Abstract

The advent of the Square Kilometre Array and its precursors marks the start of an exciting era for radio interferometry. However, with new instruments producing unprecedented quantities of data, many existing calibration algorithms and implementations will be hard-pressed to keep up. Fortunately, it has recently been shown that the radio interferometric calibration problem can be expressed concisely using the ideas of complex optimisation. The resulting framework exposes properties of the calibration problem which can be exploited to accelerate traditional non-linear least squares algorithms.

We extend the existing work on the topic by considering the more general problem of calibrating a Jones chain: the product of several unknown gain terms. We also derive specialised solvers for performing phase-only, delay and pointing error calibration. In doing so, we devise a method for determining update rules for arbitrary, real-valued parametrisations of a complex gain.

The solvers are implemented in an optimised Python package called CubiCal. CubiCal makes use of Cython to generate fast C and C++ routines for performing computationally demanding tasks whilst leveraging multiprocessing and shared memory to take advantage of modern, parallel hardware. The package is fully compatible with the measurement set, the most common format for interferometer data, and is well integrated with Montblanc - a third party package which implements optimised model visibility prediction.

CubiCal's calibration routines are applied successfully to both simulated and real data for the field surrounding source 3C147. These tests include direction-independent and direction-dependent calibration, as well as tests of the specialised solvers. Finally, we conduct extensive performance benchmarks and verify that CubiCal convincingly outperforms its most comparable competitor.

Declaration of Authorship

I, Jonathan KENYON, declare that this thesis titled, "CubiCal: A fast radio interferometric calibration suite exploiting complex optimisation" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Never be so focused on picking a lock that you forget kicking down the door is also an option."

— Mark Lawrence, Grey Sister

Acknowledgements

First and foremost, I must thank my supervisor, Prof. Oleg Smirnov for his help and contribution to this project. Similarly, I do not believe I would have reached this point without the guidance of my friend and co-supervisor Dr Trienko Grobler.

I am indebted to Dr Cyril Tasse who inspired and assisted me when this research was in its infancy. I thank Dr Simon Perkins for his assistance in integrating Montblanc and for always being available to help. I am grateful to Dr Ian Heywood for being both an unofficial tester and one of CubiCal's most active users.

The students of RATT and the members of the RARG team have my thanks for their constructive feedback. Cyndie Russeeawon deserves a special mention for catching the majority of my typos.

Of course, I must express my gratitude to my family, not only for their financial assistance, but for their unwavering support in times of crisis.

Finally, I acknowledge the financial assistance of the South African SKA Project (SKA SA) towards this research (www.ska.ac.za).

Publications

The following is a list of publications in which I was involved during this research:

- Kenyon, J. S., Smirnov, O. M., Grobler, T. L., & Perkins, S. J. (2018). "CubiCal – Fast radio interferometric calibration suite exploiting complex optimization". *Monthly Notices of the Royal Astronomical Society*, 478(2), 2399-2415.
- Grobler, T. L., Stewart, A. J., Wijnholds, S. J., Kenyon, J. S., & Smirnov, O. M. (2016). "Calibration artefacts in radio interferometry–III. Phase-only calibration and primary beam correction". *Monthly Notices of the Royal Astronomical Society*, 461(3), 2975-2992.
- Grobler, T. L., Bernardi, G., Kenyon, J. S., Parsons, A. R., & Smirnov, O. M. (2018). "Redundant interferometric calibration as a complex optimization problem". *Monthly Notices of the Royal Astronomical Society*, 476(2), 2410-2420.

Additionally, the CubiCal source code is publicly available at:

https://github.com/ratt-ru/CubiCal/.

Contents

List of Figures

List of Tables

List of Recurring Abbreviations

1GC	(First) **G**eneration **C**alibration
2GC	(Second) **G**eneration **C**alibration
3GC	(Third) **G**eneration **C**alibration
AIPS	**A**stronomical **I**mage **P**rocessing **S**ystem
ASKAP	**A**ustralian **S**quare **K**ilometre **A**rray **P**athfinder
CASA	**C**ommon **A**stronomy **S**oftware **A**pplications
CPU	**C**entral **P**rocessing **U**nit
DDE	**D**irection-**d**ependent **E**ffect
DIE	**D**irection-**i**ndependent **E**ffect
EM	**E**lectro**m**agnetic
EM	**E**xpectation-**M**aximization
FWHM	**F**ull **W**idth (at) **H**alf **M**aximum
GIL	**G**lobal **I**nterpreter **L**ock
GN	**G**auss-**N**ewton
GPU	**G**raphics **P**rocessing **U**nit
HPBW	**H**alf **P**ower **B**eam**w**idth
LM	**L**evenberg-**M**arquardt
LOFAR	**L**ow-**F**requency **Ar**ray
MeerKAT	(Meer) **K**aroo **A**rray **T**elescope
NLLS	**N**on-linear **L**east **S**quares
OOP	**O**bject-**O**riented **P**rogramming
PSF	**P**oint **S**pread **F**unction
RAM	**R**andom-**A**ccess **M**emory
RFI	**R**adio **F**requency **I**nterference
RIME	**R**adio **I**nterferometer **M**easurement **E**quation
SAGE	**S**pace-**A**lternating **G**eneralized **E**xpectation-Maximization
SKA	**S**quare **K**ilometre **A**rray
SNR	**S**ignal-to-**N**oise **R**atio
SPAM	**S**ource **P**eeling (and) **A**tmospheric **M**odeling
VLA	**V**ery **L**arge **A**rray
VLBI	**V**ery **L**ong **B**aseline **I**nterferometry

List of Recurring Symbols

N_{A}	Number of antennas
N_{BL}	Number of baselines
N_{J}	Number of Jones terms
N_{S}	Number of sources
N_{D}	Number of directions
N_{PPA}	Number of parameters per antenna
N_t	Number of integrations
N_ν	Number of channels
p, q, pq	Antenna and baseline indices
$\mathbf{J}$	Jones matrix (Chapter 2)
$\mathbf{V}$	Visibility matrix
$\mathbf{B}$	Brightness matrix
$\mathbf{G}$	Direction-independent gain/Jones term
$\mathbf{E}$	Direction-dependent gain/Jones term
$\mathcal{F}(\cdot)$	Fourier transform
r	Scalar residual
d	Scalar observed visibility
g	Scalar direction-independent gain
m	Scalar model visibility
w	Scalar weight
$\sum$	Summation
$\lvert(\cdot)\rvert$	Absolute value
$\overline{(\cdot)}$	Conjugate
$(\cdot)^T$	Transpose
$(\cdot)^{-1}$	Inverse
$(\cdot)^H$	Conjugate transpose
$(\cdot)^{-H}$	Inverse conjugate transpose
$\lvert\lvert\cdot\rvert\rvert_F^2$	Frobenius norm
$\mathbf{M}$	Model visibility matrix
$\mathbf{D}$	Observed visibility matrix
$\mathrm{vec}\,(\cdot)$	Vectorise
$\otimes$	Kronecker product
$(\cdot)^*$	Element-wise conjugation
$\mathbf{r}$	Residual vector
$\mathbf{J}$	Jacobian (Chapter 4 onwards)

$\breve{\mathbf{r}}, \breve{\mathbf{v}}, \breve{\mathbf{g}}$	Augmented vectors (Section 4.3)
$\text{diag}(\cdot)$	Diagonal entries
$\mathcal{L}$	Left multiply operator (Subsection 4.3.2)
$\mathcal{R}$	Right multiply operator (Subsection 4.3.2)
$\mathbb{W}$	Isomorphism (Subsection 4.3.2)
$\vec{\mathbf{r}}$	Nested vector (Subsection 4.3.2)
$(\cdot)^\dagger$	Element-wise conjugate transpose (Subsection 4.3.2)
$\prod$	Product
$\mathbf{H}$	Hessian
$\tilde{\mathbf{H}}$	Approximate Hessian
$\odot$	Hadamard product
δ	Dirac delta
$\mathbb{R}\text{e}(\cdot)$	Real component
$\mathbb{I}\text{m}(\cdot)$	Imaginary component
$\hat{\mathbf{A}}$	Correlation matrix (Section 6.3)
$\diamond$	Aligned product (Section 6.3)

Preface

To say that we are in an exciting era for both radio interferometry and the field of astronomy in general is a gross understatement. With the advent of the Square Kilometre Array (SKA) and its precursors, MeerKAT and HERA (Hydrogen Epoch of Reionization Array) in South Africa and ASKAP (Australia SKA Pathfinder) and the MWA (Murchison Widefield Array) in Australia, we will have an unprecedented capacity to probe the radio universe. This will be exploited to study the most elusive of radio phenomena and attempt to answer questions regarding the origins of the universe.

Similarly ambitious projects are in the works at optical wavelengths, including (but not limited to) the LSST (Large Synoptic Survey Telescope) and E-ELT (European Extremely Large Telescope), both of which will be constructed in Chile, and the TMT (Thirty Meter Telescope) to be constructed in Hawaii. While each of these instruments has its own science goals, the ultimate result will be a proliferation of high quality optical data.

It is not only astronomy in the electromagnetic spectrum that is contributing to our ever improving picture of the universe. LIGO (Laser Interferometer Gravitational-Wave Observatory) recently made the first successful direct detection of gravitational waves, originating from the merger of two black holes.

Taking all of the above into consideration, the future certainly seems bright for astronomers and astronomy the world over. However, it is easy to forget that behind any sufficiently ambitious project is a frankly daunting amount of work. In fact, the techniques and technologies required often don't exist when the projects are first conceived. This results in behind-the-scenes innovation which, while less glamorous than constructing a telescope atop a mountain or launching a satellite, is no less important to the progress of science.

The SKA (and, to a lesser extent, its precursors) poses some particularly difficult technical challenges. Setting aside the practical difficulties of its construction, perhaps the largest problem is simply how to handle the staggering amount of data it will produce. Not only does this include (relatively) straightforward questions regarding how said data will be stored, but also how it will be processed. This processing will, by necessity, include calibration (correction of errors in the data). Calibration has traditionally been a computationally demanding task and it is on this topic that this document aims to make a contribution.

We begin in Chapter 1 with a very brief introduction to radio astronomy, focussing on the most conventional observables, antennas, and primary beams. This leads us neatly into Chapter 2, in which we present the basics of interferometry and introduce the RIME (Radio Interferometer Measurement Equation), which underpins the majority of this work.

Chapter 3 provides an overview of calibration in the context of radio interferometry, and describes the origins of the self-calibration methodology. This is followed by a discussion of

several of the most common contemporary calibration algorithms and software packages.

In Chapter 4, we present the mathematical background required by our later derivations. This includes a description of calibration as a conventional non-linear least squares (NLLS) problem in addition to a discussion of complex optimisation. Much of the notation required to understand subsequent chapters is included here.

Chapter 5 is the most important chapter as it contains our derivations of update rules for performing several different types of calibration using complex optimisation. It is these update rules upon which the CubiCal software package is based; a discussion of its implementation is the topic of Chapter 6.

Having demonstrated how CubiCal works, Chapters 7 and 8 test CubiCal's ability to calibrate simulated and real data respectively. Chapter 9 presents the results of some simple benchmarks which demonstrate CubiCal's competitive performance.

Finally, the document concludes with a summary of our findings and some comments on the directions in which CubiCal may evolve in the future.

Chapter 1

A Brief Introduction to Radio Astronomy

Optical astronomy has always been the most intuitive branch of astronomy. In fact, for a time, it was the only branch of astronomy. This is due, at least in part, to the fact that optical astronomy deals with electromagnetic (EM) radiation in the visible portion of the EM spectrum - what is commonly called visible light. This association with vision and sight makes it easy to think of optical telescopes as simply being more sensitive eyes - eyes which can detect far fainter emission, far further away. However, spanning wavelengths between just 390nm and 700nm (frequencies between 430 THz and 770 THz), visible light is only a small portion of the EM spectrum. Consequently, important and powerful though it is, optical astronomy provides only a part of the complete EM picture of the universe.

Filling in this picture requires astronomy at different wavelengths including gamma-ray, X-ray and ultra-violet astronomy at higher frequencies and infra-red and radio astronomy at lower frequencies. It is the last of these, radio astronomy, that is of particular interest to us.

The first observations of radio emission from a celestial source were made by Karl G. Jansky in 1932 (Jansky 1932). Whilst performing an unrelated investigation for Bell Telephone Laboratories, he observed emission of unknown origin at 20.5 MHz. He subsequently showed (Jansky 1933) that it must originate from the centre of the Milky Way. His research, however, went largely unnoticed and several years passed before Grote Reber successfully mapped the Milky Way at 160 MHz (Reber 1940) using a parabolic antenna which he had constructed in his garden. Thus, the field of radio astronomy was truly born.

Terrestrial radio astronomy (measurements are made from within the Earth's atmosphere) is fundamentally limited to the frequency range between 15 MHz and 1.5 THz (Wilson et al. 2009). The high frequency cut-off is caused by vibrating molecules in the Earth's atmosphere (such as CO_2, O_2, and H_2O) which absorb incoming high frequency emission. The low frequency cut-off is caused by a combination of absorption, refraction and reflection by the ionosphere.

Many classes of astronomical sources emit radiation within this band via almost equally numerous emission mechanisms. Some such sources include supernova remnants, pulsars, galaxy clusters, radio halos and the cosmic microwave background (CMB). This is by no means an exhaustive list - the important point is that there is a wealth of science which can be performed at radio wavelengths.

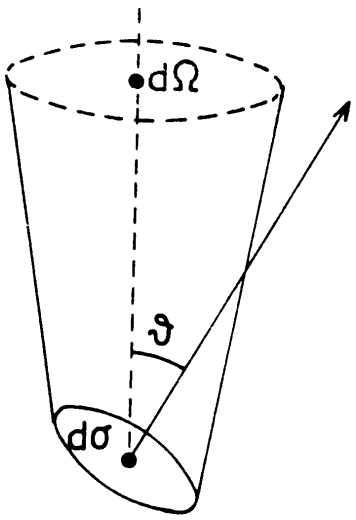

FIGURE 1.1: Relationship between infinitesimal solid angle $\mathrm{d}\Omega$, infinitesimal surface area $\mathrm{d}\sigma$ and ϑ, the angle between the line normal to $\mathrm{d}\sigma$ and the centre of $\mathrm{d}\Omega$. Diagram from Wilson et al. (2009).

1.1 Fundamental observables

It is important to understand precisely what a radio telescope measures. In order to do so, let us consider an observer intercepting radio emission from an astronomical source (see Figure 1.1). This situation can be described by the following expression (Wilson et al. 2009):

$$\mathrm{d}P = I_\nu \cos(\vartheta)\mathrm{d}\Omega\mathrm{d}\sigma\mathrm{d}\nu, \tag{1.1}$$

where $\mathrm{d}P$ is the infinitesimal power (W) intercepted by infinitesimal surface area $\mathrm{d}\sigma$ (m^2) in the infinitesimal bandwidth $\mathrm{d}\nu$ (Hz). The angle ϑ is the angle between a line normal to $\mathrm{d}\sigma$ and the centre of $\mathrm{d}\Omega$, the infinitesimal solid angle (sr) subtended by the source of the radio emission. I_ν is then defined as the brightness, intensity or specific intensity:

$$I_\nu \equiv \frac{\mathrm{d}P}{\cos(\vartheta)\mathrm{d}\Omega\mathrm{d}\sigma\mathrm{d}\nu}, \tag{1.2}$$

and has units of W m^{-2} sr^{-1} Hz^{-1}. One of the goals of radio astronomy is to map this quantity over the entire sky at all radio frequencies. Note that the above expressions are only valid when the assumptions of geometrical optics (Born and Wolf 2013) hold i.e. that the source of the emission is sufficiently far away that the incident radiation can be considered to travel in straight lines (rays).

A related quantity which is often of interest is the flux density. This quantity is used for discrete sources i.e. relatively compact sources which subtend a known solid angle. By integrating Equation 1.1 over the entire solid angle subtended by the source (Ω_s), we obtain the flux density:

$$S_\nu = \int_{\Omega_s} I_\nu \cos(\vartheta)\mathrm{d}\Omega, \tag{1.3}$$

the units of which are W m^{-2} Hz^{-1}. The flux densities of radio sources are usually very small and are often expressed in units of Jansky (Jy):

$$1\text{Jy} = 1 \times 10^{-26}\text{Wm}^{-2}\text{Hz}^{-1}. \tag{1.4}$$

There are a couple of important distinctions to be drawn between the brightness and the flux density. First and foremost, unlike flux density, brightness is a property intrinsic to the source - in the absence of other factors, it does not vary with distance. A concomitant of this is that the brightness is the same both at the source and at the detector. Flux density does not share these properties due to its dependence on the solid angle subtended by the source which is itself dependent on the distance between the source and the observer.

1.2 Antennas

Having established our observables it is also pertinent to describe the method by which we measure them - antennas. An antenna can be described simply as the portion of a radio receiver which converts radio emission propagating though space into electrical signals propagating through wires (Graf 1999). It is these electrical signals that we use to extract information about the radio sky. What follows is a somewhat cursory explanation of some of the important properties of antennas. Both Wilson et al. (2009) and Balanis (2005) provide far more detailed accounts.

Perhaps the most important property of an antenna within the scope of this document is its radiation pattern. The radiation pattern of an antenna describes its response to an incoming signal as a function of direction (spatial coordinates). Figure 1.2 is an example of a radiation pattern.

Unlike a theoretically perfect, isotropic antenna, a real antenna is not equally sensitive in all directions. This typically manifests as the multi-lobe structure apparent in Figure 1.2. Practically, this means that the position of a source with respect to the radiation pattern will determine the degree to which that source's signal is attenuated.

Balanis (ibid.) makes use of two different terms when describing radiation patterns:

- Field pattern: the magnitude of the electric or magnetic field as a function of direction.
- Power pattern: the square of the magnitude of the electric or magnetic field as a function of direction.

In radio astronomy, the terms radiation pattern and beam are used somewhat interchangeably. Consequently, the second of these terms is often referred to as the power beam. While both of these quantities only consider the magnitude of the radiation pattern, it is important to remember that the beam of a real antenna will also affect the phase of incoming signals. We refer to the combined amplitude and phase effects of the antenna radiation pattern as the primary beam of the antenna.

It is often necessary to quantify the size of the radiation pattern, at least in terms of its major lobe (often called the main beam). This is done using the full width at half-maximum

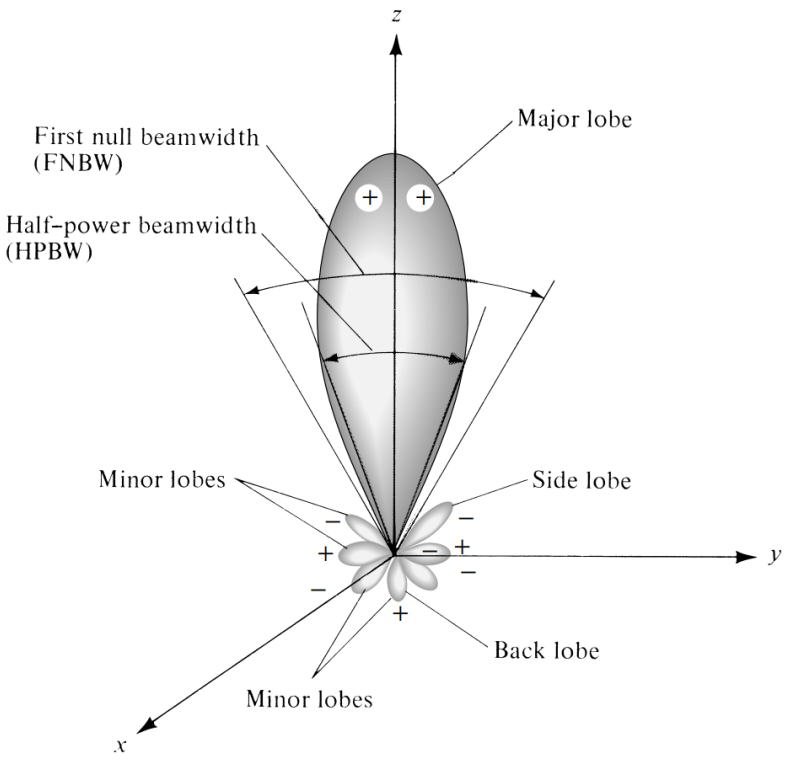

FIGURE 1.2: An example radiation pattern for an antenna pointing along the z-axis. Diagram from Balanis (2005).

(FWHM) or half-power beam width (HPBW), both of which are measures of angular extent of the beam between between the points at which the major lobe falls to half of its maximum value.

A second important property of antennas is the role that aperture dimension plays in determining the angular resolution of the instrument. Angular resolution is a measure of an instrument's sensitivity to detail. The angular resolution of a telescope (R) is inversely proportional to its diameter, D, and directly proportional to the wavelength, λ, at which the telescope observes:

$$R \propto \frac{\lambda}{D}. \tag{1.5}$$

This expression exposes a fundamental limitation of single dish radio astronomy. Radio waves have large wavelengths and obtaining high angular resolution requires very large telescopes. Constructing telescopes of this size is usually impractical, if not impossible. Fortunately, radio interferometry provides us with an alternative.

Chapter 2

Radio Interferometry

A complete and exhaustive explanation of the field of radio interferometry would require several volumes of text. Fortunately, there exists a mathematically concise, intuitive description called the radio interferometer measurement equation (RIME) which will be sufficient for our purposes. Originally proposed by Hamaker et al. (1996) using a slightly counter-intuitive form involving 4×4 matrices, subsequent developments (Hamaker 2000) led to the emergence of an equivalent formalism using 2×2 matrices. It is this version of the RIME that underpins much of the work herein.

However, before pursuing the RIME and the associated mathematics further, it is pertinent to describe a radio interferometer and the principles on which it operates. Like the Michelson interferometer (Michelson and Morley 1887), a radio interferometer measures the interference pattern between electromagnetic waves (specifically at radio wavelengths) in order to extract information about the source of that emission.

An example of a simple, two-element interferometer is provided in Figure 2.1. The elements in question are antennas (p and q) and, in general, a real instrument will have far more than two. However, large antenna arrays can always be treated as multiple two-element interferometers or, as they are commonly referred to in this case, baselines. For an array composed of N_{A} antennas, the number of unique baselines, N_{BL} is given by (Thompson et al. 2017):

$$N_{\mathrm{BL}} = \frac{N_{\mathrm{A}}(N_{\mathrm{A}} - 1)}{2}. \tag{2.1}$$

The angular resolution of an array is proportional to the longest of these baselines. We note in passing that both N_{A} and N_{BL} will feature prominently throughout this document - they are often used to describe the shape and size of the problems which we will be solving.

We are now in a good position to return to the basics of the RIME formalism. The derivation which we will present here follows Smirnov (2011a), which summarises the preceding work on the topic.

Let us presume that we are observing a quasi-monochromatic (narrow-bandwidth) source. A complex vector $\mathbf{e}$ can be used to describe the emission from this source at a given instant and position. If we assume an orthonormal coordinate system, e.g. xyz where the z-axis points from the antenna to the source, the signal vector $\mathbf{e}$ is given by:

$$\mathbf{e} = \begin{bmatrix} e_x \\ e_y \end{bmatrix}, \tag{2.2}$$

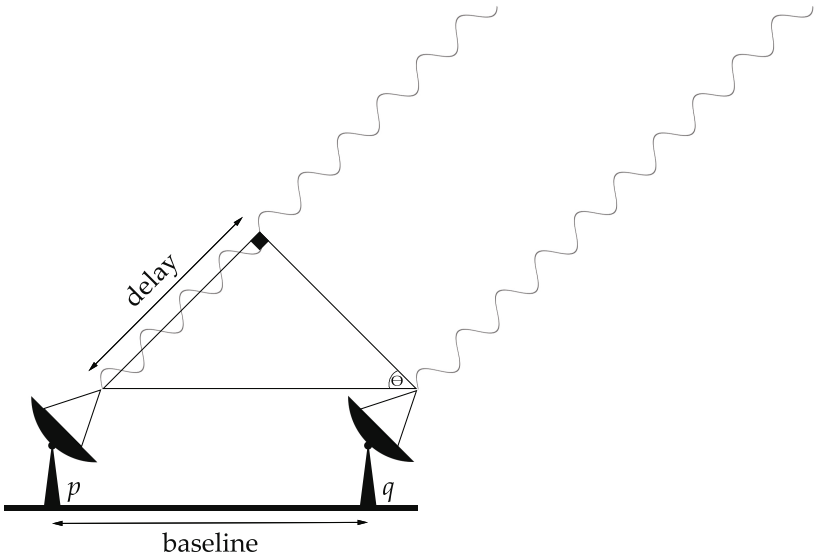

FIGURE 2.1: A simple, two-element interferometer.

noting that the e_z component is zero under the assumption that we are observing a plane wave. This is true for any sufficiently distant source.

We assume that any effect on the signal can be represented by a multiplication of $\mathbf{e}$ by a 2×2 complex matrix $\mathbf{J}$. This is equivalent to an assumption of linearity. The matrix $\mathbf{J}$ is referred to as a Jones matrix (Jones 1941) and Smirnov (2011a) coins the term Jones chain to describe the product of several Jones matrices. These Jones chains can be used to describe any number of sequential effects on the signal. Naturally, we are free to group and collapse the elements of a Jones chain as necessary:

$$\mathbf{e}' = \mathbf{J}^{(1)}\mathbf{J}^{(2)}\dots\mathbf{J}^{(n)}\mathbf{e} = \mathbf{J}\mathbf{e}. \tag{2.3}$$

The Jones terms closest to $\mathbf{e}$ affect the signal first. We, however, number them in reverse order - this is more organic to read and will make some of equations which we will present less confusing.

The signal vector is converted into complex-valued voltages by the antenna feeds. The most common feeds are linear (identified by X and Y) and circular (identified by L and R). We also assume that the conversion can be represented by a Jones matrix, i.e. the voltages are linear with respect to the signal vector:

$$\mathbf{v} = \begin{bmatrix} v^X \\ v^Y \end{bmatrix} = \mathbf{J}\mathbf{e}. \tag{2.4}$$

This expression neatly summarises the relationship between the measured voltage and the signal vector. The so-called total Jones matrix represents the cumulative result of all propagation and electronic effects.

Thus far, we have only considered a single antenna. However, a single baseline is made up of two antennas (see Figure 2.1), p and q. Each antenna produces its own voltage vector

which we will notate as $\mathbf{v}_p$ and $\mathbf{v}_q$. These voltage vectors are correlated to produce four, pairwise correlations which can be arranged into a visibility matrix:

$$\mathbf{V}_{pq} = 2 \begin{bmatrix} \langle v_p^X \bar{v}_q^X \rangle & \langle v_p^X \bar{v}_q^Y \rangle \\ \langle v_p^Y \bar{v}_q^X \rangle & \langle v_p^Y \bar{v}_q^Y \rangle \end{bmatrix}, \tag{2.5}$$

where $\langle \cdot \rangle$ represents averaging over the correlator channel and integration bin and $\bar{(\cdot)}$ denotes conjugation. The factor of two is included here in order to make a later definition neater.

Equation 2.5 can also be written as the product of two voltage vectors:

$$\mathbf{V}_{pq} = 2 \left\langle \begin{bmatrix} v_p^X \\ v_p^Y \end{bmatrix} \begin{bmatrix} \bar{v}_q^X & \bar{v}_q^Y \end{bmatrix} \right\rangle = 2 \left\langle \mathbf{v}_p \mathbf{v}_q^H \right\rangle, \tag{2.6}$$

with $(\cdot)^H$ representing the conjugate/Hermitian transpose. Equations 2.4 through 2.6 are expressed in terms of linear feed components but the same expressions hold for circular feeds if we replace X with L and Y with R.

Equations 2.4 and 2.6 can be combined, noting that each antenna has its own total Jones matrix ($\mathbf{J}_p$ and $\mathbf{J}_q$) which describes effects on the signal received at the associated antenna:

$$\mathbf{V}_{pq} = 2 \left\langle \mathbf{J}_p \mathbf{e} (\mathbf{J}_q \mathbf{e})^H \right\rangle = 2 \left\langle \mathbf{J}_p (\mathbf{e}\mathbf{e}^H) \mathbf{J}_q^H \right\rangle. \tag{2.7}$$

Jones terms which are constant over the averaging interval do not need to be included in the average. We will assume that this is the case and rewrite Equation 2.7 as:

$$\mathbf{V}_{pq} = 2\mathbf{J}_p \left\langle \mathbf{e}\mathbf{e}^H \right\rangle \mathbf{J}_q^H = 2\mathbf{J}_p \begin{bmatrix} \langle e_x \bar{e}_x \rangle & \langle e_x \bar{e}_y \rangle \\ \langle e_y \bar{e}_x \rangle & \langle e_y \bar{e}_y \rangle \end{bmatrix} \mathbf{J}_q^H. \tag{2.8}$$

The components of the matrix appearing in Equation 2.8 are particularly interesting; they are associated with the Stokes parameters which describe the polarisation properties of an electromagnetic signal (Born and Wolf 2013; Thompson et al. 2017). In fact, it is possible to prove that (Hamaker et al. 1996):

$$2 \begin{bmatrix} \langle e_x \bar{e}_x \rangle & \langle e_x \bar{e}_y \rangle \\ \langle e_y \bar{e}_x \rangle & \langle e_y \bar{e}_y \rangle \end{bmatrix} = \begin{bmatrix} I + Q & U + iV \\ U - iV & I - Q \end{bmatrix} = \mathbf{B}. \tag{2.9}$$

The newly introduced matrix, $\mathbf{B}$, is termed the brightness matrix and, following its definition, the first and simplest version of the RIME emerges from Equation 2.8:

$$\mathbf{V}_{pq} = \mathbf{J}_p \mathbf{B} \mathbf{J}_q^H. \tag{2.10}$$

This is the RIME for a single point source which neatly relates the visibility observed on baseline pq to the Stokes parameters of the source and the Jones terms associated with antennas p and q.

The RIME is often used in its appropriately named onion form. The reason for the name becomes apparent when we move from using a single, total Jones term per antenna back to

the multiple Jones terms present in Equation 2.3:

$$\mathbf{V}_{pq} = \mathbf{J}_p \mathbf{B} \mathbf{J}_q^H = \mathbf{J}_p^{(1)}(\dots(\mathbf{J}_p^{(n-1)}(\mathbf{J}_p^{(n)}\mathbf{B}\mathbf{J}_q^{(n)H})\mathbf{J}_q^{(n-1)H})\dots)\mathbf{J}_q^{(1)H}. \tag{2.11}$$

These Jones terms generally belong to one of the following categories: scalar, diagonal, rotation or general 2×2. We will not adopt unique notation for each of these categories; we will note in the text the category to which a term belongs.

Scalar Jones terms are the simplest of the Jones terms and represent effects which affect both components of the signal vector equally. Importantly, scalar Jones terms commute with all other Jones terms. For a constant c, a scalar Jones term has the following form:

$$\mathbf{J}_{\text{scalar}} = c \equiv \begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix} = c \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{2.12}$$

Diagonal Jones terms commute amongst themselves and with scalar Jones terms but never with anything else. They are used to describe effects which are unique per component of the signal vector but which do not involve the mixing of those components. Note that a change in coordinate system can change the diagonality of a Jones term. For two constant values c_1 and c_2, a diagonal matrix has the following form:

$$\mathbf{J}_{\text{diagonal}} = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix}, \tag{2.13}$$

where we note that if $c_1 = c_2$ we are once again left with a scalar matrix.

Unlike diagonal matrices, rotation matrices allow intermixing between the components of the signal vector. Like diagonal matrices, they commute only with scalar matrices and amongst themselves. Again, like diagonality, a Jones term being a rotation matrix depends on our choice of coordinate system. As an example, a 2×2 rotation matrix in an xy-frame has a diagonal matrix representation in a circular frame. A rotation matrix in an xy-frame, given rotation angle ϕ, has the following form:

$$\mathbf{J}_{\text{rotation}} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}. \tag{2.14}$$

The last category of Jones term, general 2×2, also allows intermixing of the components of the signal vector. Jones terms of this form do not commute with anything other than scalar Jones terms. We do not provide an example here as these terms will appear regularly in later sections.

At this juncture, it is necessary to begin defining some specific Jones terms. The first of these is a scalar Jones term corresponding to the geometric delay in the signals received by antennas p and q (see Figure 2.1). This term is present even in an observation otherwise free of corruptions.

The geometric delay manifests as a phase difference between the signals received at antennas p and q. Generally, we minimise this phase difference in a particular direction, effectively

pointing our array at the so-called phase centre. This is usually performed in the correlator, where an additional delay can be inserted for this purpose.

A full description of the coordinate systems used in radio interferometry is outside the scope of this work. Consequently, we will adopt the conventions and notation of Thompson et al. (2017). Let us assume that antenna p is located at uvw-coordinates $\mathbf{u}_p = [u_p, v_p, w_p]$. The phase difference relative to an arbitrarily chosen $\mathbf{u}_0 = [0, 0, 0]$ for a signal originating from direction $\boldsymbol{\sigma}$ is then given by:

$$\kappa_p = 2\pi\lambda^{-1}(u_p l + v_p m + w_p(n-1)). \tag{2.15}$$

Here l, m and $n = \sqrt{1-l^2-m^2}$ are direction cosines of $\boldsymbol{\sigma}$ (cosines of the angles between the vector and its associated coordinate axes) and λ is the wavelength of the signal. The uvw-coordinates are usually given in units of wavelength, which removes the need for the λ^{-1} term. Note that the minus one that appears in the final term is a result of fringe stopping (ibid.). The scalar $\mathbf{K}$ Jones term (as described in Noordam (1996)) is then defined as:

$$\mathbf{K}_p = \begin{bmatrix} e^{-i\kappa_p} & 0 \\ 0 & e^{-i\kappa_p} \end{bmatrix} = e^{-i\kappa_p} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \equiv e^{-i\kappa_p} = e^{-2\pi i(u_p l + v_p m + w_p(n-1))}. \tag{2.16}$$

The RIME of an uncorrupted point source is consequently given by:

$$\mathbf{V}_{pq} = \mathbf{K}_p \mathbf{B} \mathbf{K}_q^H = \mathbf{B} e^{-2\pi i((u_p-u_q)l + (v_p-v_q)m + (w_p-w_q)(n-1))}. \tag{2.17}$$

The right hand side of Equation 2.17 shows that the visibility observed on baseline pq is a function of its uvw-coordinates, $\mathbf{u}_{pq} = \mathbf{u}_p - \mathbf{u}_q = [u_p - u_q, v_p - v_q, w_p - w_q]$. In order to remain consistent with Smirnov (2011a), the visibility matrix produced by Equation 2.17 is referred to as the source coherency and notated as $\mathbf{X}_{pq}$. It is a complex 2×2 matrix and can be thought of as the visibility that would be measured by an interferometer (baseline) in the absence of any corruption.

Of course, in reality there are always corruptions which creep into the signal path. The specifics of these errors will be discussed in due course, but for now we can include them by adding an extra term to our Jones chain:

$$\mathbf{J}_p = \mathbf{G}_p \mathbf{K}_p, \tag{2.18}$$

where $\mathbf{G}_p$ is a single complex-valued gain which incorporates all corrupting effects. This makes it simple to write down the RIME for a corrupted point source as:

$$\mathbf{V}_{pq} = \mathbf{G}_p \mathbf{K}_p \mathbf{B} \mathbf{K}_q^H \mathbf{G}_q^H = \mathbf{G}_p \mathbf{X}_{pq} \mathbf{G}_q^H. \tag{2.19}$$

Of course, we often wish to consider a sky containing more than a single point source. Each source will be subject to its own set of propagation effects which can be described by a

Jones term/chain. For N_{S} point sources, our generic RIME expression is:

$$\mathbf{V}_{pq} = \sum_{s=1}^{N_{\mathrm{S}}} \mathbf{J}_{sp}\mathbf{B}_s\mathbf{J}_{sq}^H. \tag{2.20}$$

We once again consider that each $\mathbf{J}$ is a Jones chain comprised of several Jones terms which do not generally commute. We adopt the following generalised chain, noting that terms on the left are considered to arise at the antenna end of the propagation path and terms on the right originate at the source end:

$$\mathbf{J}_{sp} = \mathbf{G}_p\mathbf{E}_{sp}\mathbf{K}_{sp}. \tag{2.21}$$

In Equation 2.21, $\mathbf{G}_p$ represents the combined effect of all the source-independent terms and $\mathbf{E}_{sp}$ is the equivalent for source-dependent terms. $\mathbf{K}_{sp}$ is a scalar, source-dependent phase term which arises from the geometric delays. Due to its scalar nature, $\mathbf{K}_{sp}$ can appear anywhere in the chain but is usually placed on the source end of the chain. Incorporating Equation 2.21 into Equation 2.20 we obtain:

$$\mathbf{V}_{pq} = \mathbf{G}_p \left(\sum_{s=1}^{N_{\mathrm{S}}} \mathbf{E}_{sp}\mathbf{K}_{sp}\mathbf{B}_s\mathbf{K}_{sq}^H\mathbf{E}_{sq}^H \right) \mathbf{G}_q^H = \mathbf{G}_p \left(\sum_{s=1}^{N_{\mathrm{S}}} \mathbf{E}_{sp}\mathbf{X}_{spq}\mathbf{E}_{sq}^H \right) \mathbf{G}_q^H. \tag{2.22}$$

At this point, following Smirnov (2011a), we introduce two new terms which we will use to describe the components of Equation 2.22: direction-independent effects (DIEs) and direction-dependent effects (DDEs). $\mathbf{G}$ terms are associated with DIEs and $\mathbf{E}$ terms are associated with DDEs. Note that direction-dependence is equivalent to source-dependence in the limit where each source is considered a separate direction. The converse, that we may define a single "direction" as containing the contribution of more than one source (sometimes referred to as "clustering"), will prove useful later.

The version of the RIME present in Equation 2.22 is almost sufficient to begin a discussion of calibration. However, in the interest of presenting a more complete picture of radio interferometry, we will continue just a little further.

The principal limitations of Equation 2.22 are that it cannot capture the behaviour of extended emission as it is inherently discretised. For a continuous brightness distribution, $\mathbf{B}(\boldsymbol{\sigma})$, where $\boldsymbol{\sigma}$ is a direction vector, the visibility measured by baseline pq is given by:

$$\mathbf{V}_{pq} = \int_{4\pi} \mathbf{J}_p(\boldsymbol{\sigma})\mathbf{B}(\boldsymbol{\sigma})\mathbf{J}_q^H(\boldsymbol{\sigma})d\Omega, \tag{2.23}$$

where $\mathbf{J}_p(\boldsymbol{\sigma})$ describes any effects along the signal path in direction $\boldsymbol{\sigma}$. In general, performing this integration over the unit sphere is very difficult. Instead, a projection onto the lm-plane tangent to the field centre is used, recasting Equation 2.23 as:

$$\mathbf{V}_{pq} = \iint_{lm} \mathbf{J}_p(\mathbf{l})\mathbf{B}(\mathbf{l})\mathbf{J}_q^H(\mathbf{l})\frac{dldm}{n}, \quad \text{where} \quad \mathbf{l} = [l, m]. \tag{2.24}$$

The Jones terms, as seen in the case of multiple point sources, can be split into a direction-independent component ($\mathbf{G}$), a direction-dependent component ($\mathbf{E}^{'}$) and the delay component ($\mathbf{K}$). Thus, the Jones chain can be written as:

$$\mathbf{J}_p(\mathbf{l}) = \mathbf{G}_p\mathbf{E}_p^{'}(\mathbf{l})\mathbf{K}_p(\mathbf{l}) = \mathbf{G}_p\mathbf{E}_p^{'}(\mathbf{l})e^{-2\pi i(u_p l + v_p m + w_p(n-1))}. \tag{2.25}$$

Substituting this expression back into Equation 2.24 and noting that we can combine the scalar $\mathbf{K}$ terms, we obtain:

$$\mathbf{V}_{pq} = \mathbf{G}_p \left(\iint\limits_{lm} \frac{1}{n}\mathbf{E}_p^{'}\mathbf{B}\mathbf{E}_q^{'H} e^{-2\pi i(u_{pq}l + v_{pq}m + w_{pq}(n-1))} dldm \right) \mathbf{G}_q^H. \tag{2.26}$$

A close inspection of this expression reveals striking similarities between it and a two-dimensional Fourier transform. In fact, it is only the presence of the w-term (also known as the non-coplanarity term) which prevents it from being precisely that. In an effort to side-step this problem, we split the w-term into per-antenna components:

$$\mathbf{W}_p = \frac{1}{\sqrt{n}} e^{-2\pi i w_p(n-1)}. \tag{2.27}$$

These $\mathbf{W}$ terms are scalar and direction-dependent. Consequently, we can combine them with our already defined $\mathbf{E}^{'}$ terms and define $\mathbf{E}_p = \mathbf{E}_p^{'}\mathbf{W}_p$. Equation 2.26 can then be rewritten as a two-dimensional Fourier transform:

$$\mathbf{V}_{pq} = \mathbf{G}_p \left(\iint\limits_{lm} \mathbf{B}_{pq} e^{-2\pi i(u_{pq}l + v_{pq}m)} dldm \right) \mathbf{G}_q^H, \quad \text{where} \quad \mathbf{B}_{pq} = \mathbf{E}_p\mathbf{B}\mathbf{E}_q^H. \tag{2.28}$$

This result shows us that each baseline measures the Fourier transform of the sky brightness distribution. However, due to the direction-dependent terms present in $\mathbf{B}_{pq}$, each baselines effectively sees a different sky. Only in the case where $\mathbf{E}_p(\mathbf{l}) \equiv \mathbf{E}(\mathbf{l})$ do all the baselines see the same apparent sky. In this instance, the apparent sky is the true sky modified by the primary beam of the array:

$$\mathbf{B}_{pq}(\mathbf{l}) \equiv \mathbf{B}_{\text{APP}}(\mathbf{l}) = \mathbf{E}(\mathbf{l})\mathbf{B}(\mathbf{l})\mathbf{E}^H(\mathbf{l}), \tag{2.29}$$

and we may write the full-sky RIME as:

$$\mathbf{V}_{pq} = \mathbf{G}_p\mathcal{F}(\mathbf{B}_{\text{APP}})\mathbf{G}_q^H = \mathbf{G}_p\mathbf{X}(u_{pq}, v_{pq})\mathbf{G}_q^H = \mathbf{G}_p\mathbf{X}_{pq}\mathbf{G}_q^H, \tag{2.30}$$

where $\mathcal{F}(\cdot)$ denotes a two-dimensional Fourier transform.

There is a quietly remarkable result lurking in Equation 2.28. It is a version of the van Cittert-Zernike theorem (van Cittert 1934; Zernike 1938); a theorem upon which much of radio interferometry is based. Here, it is shown to appear within the RIME framework.

Before concluding our discussion of the basic RIME, we must draw attention to the fact that, like both Hamaker et al. (1996) and Smirnov (2011a), we have presented a noise-free

version of the RIME. In reality, each entry of the visibility matrix $\mathbf{V}_{pq}$ is accompanied by additive, uncorrelated Gaussian noise in both the real and imaginary parts (Thompson et al. 2017).

Chapter 3

Calibration

Calibration generally refers to the process of configuring an instrument to provide measurements which are known to be correct, at least up to a specified tolerance. In the context of radio interferometry, this is usually achieved by observing a source or sources with known parameters and determining the value of some complex-valued gain term (e.g. **G** or **E** in the preceding chapter) which will make the data consistent with those parameters.

Radio interferometric calibration is often grouped into three categories: first, second and third generation (referred to as 1GC, 2GC and 3GC respectively). These classifications were originally introduced in Noordam and Smirnov (2010) and conveniently divide the wide field of calibration into three, more manageable pieces. Referring to these as generations is highly appropriate; 1GC techniques predate both 2GC and 3GC and 2GC techniques predate 3GC.

This chapter will describe the aforementioned categories before moving on to a discussion of some of the most commonly used algorithms. Finally, some mention will be made of the software tools implementing these approaches.

3.1 First Generation Calibration (1GC)

First generation calibration techniques (sometimes referred to as external or primary calibration) involve solving for gains on calibrator fields (Fomalont and Perley 1999). These fields contain sources with known parameters (i.e. source position, source shape, polarised flux, unpolarised flux and spectral index). Using these parameters, it is possible to solve for gains on the calibrator field which can subsequently be transferred to the target field. As a result, these gains are often referred to as transfer solutions.

Observations of these calibrator fields are usually interspersed between observations of the target, under the assumption that variations in the calibrator gain solutions will approximately track changes in the gains of the target field. This assumption places strict requirements on calibrator sources based on the type of calibration with which they are associated:

- Absolute flux calibration: This type of calibration is undertaken to ensure that that the scale of the observed visibilities is correct, i.e. to ensure that the sources have the correct flux values. Performing absolute flux calibration requires a very bright, invariant calibrator which is either point-like or has a sufficiently accurate model.

- Bandpass calibration: Bandpass calibrators are observed in order to determine the frequency response of the instrument. Bandpass calibration requires an exceptionally bright, unpolarised, invariant point-like or accurately modelled source with known spectral behaviour. Sources satisfying these requirements are few and far between and, consequently, they may be very far away from the target field.

- Delay calibration: Phase delay errors manifest as a linear ramp in the bandpass i.e. a slope in frequency. Delay calibration is used to correct this behaviour. This error is expected to be very stable (for modern, digital systems) and can be fit once using all the data. Delay calibration can be performed using the same source as for bandpass calibration (satisfying the same requirements) and is usually performed prior to bandpass calibration.

- Gain calibration: Gain calibrators are observed the most regularly. The complex-valued, per-antenna gains solved for using these calibrator observations are interpolated and applied to the target field, under the assumption that they will track the evolution of local effects, such as the atmosphere. Consequently, gain calibrators need to be close to the target field and sufficiently bright to constrain a gain solution.

The manner in which gain solutions are obtained is not actually set in stone. Several methods may be used, and 1GC describes a family of techniques rather than specific mathematical methods. This is actually true for all generations of calibration; it may be done in a multitude of ways but the "flavour" of each generation is unique. We will present some of the most common mathematical methods in the following chapter.

3.2 Second Generation Calibration (2GC)

Second generation calibration is what is commonly known as self-calibration or self-cal. Its history is rich and its emergence can be attributed to a number of factors. Ekers (1984) provides a detailed overview of its early history, a truncated summary of which follows. We also discuss the basic workings of self-calibration and briefly introduce some of the more common algorithms.

3.2.1 Origins and History

The beginnings of self-calibration can be traced to the 1950s and the work of Jennison (1954, 1958) who showed that, using an interferometer, it is possible to obtain a largely error free measurement of the phase contribution of a source brightness distribution. This can be done by considering the so-called *closure phase* (a term coined by Rogers et al. (1974)).

A closure phase measurement is formed by considering phase contributions around a closed baseline loop. The simplest example is a three-element interferometer. In the scalar case, each baseline will measure the following:

$$v'_{pq} = g_p v_{pq} \bar{g} = |g_p| e^{i\theta_p} |v_{pq}| e^{i\phi_{pq}} |g_q| e^{-i\theta_q}, \tag{3.1}$$

where v'_{pq} is the observed visibility (subject to errors), v_{pq} is the uncorrupted visibility, and g_p and g_q are antenna-based gain errors. Each term can be decomposed into an amplitude and phase component. The amplitudes are denoted by $|(\cdot)|$. The phases associated with the gain errors are given by θ_p and θ_q, and the the phase of the uncorrupted visibility is given by ϕ_{pq}.

If we consider only the phases of Equation 3.1, we obtain:

$$\phi'_{pq} = \theta_p + \phi_{pq} - \theta_q + \epsilon_{pq}, \tag{3.2}$$

where ϕ'_{pq} is the measured phase and ϵ_{pq} is some error due to the noise. Let us label the three antennas in our interferometer 1,2 and 3. The closure phase is then formed by accumulating the contribution of Equation 3.2 around a closed baseline loop:

$$\begin{aligned}
\psi_{123} &= \phi'_{12} + \phi'_{23} + \phi'_{31} \\
&= \phi'_{12} + \phi'_{23} - \phi'_{13} \\
&= \theta_1 + \phi_{12} - \theta_2 + \epsilon_{12} + \theta_2 + \phi_{23} - \theta_3 + \epsilon_{23} - (\theta_1 + \phi_{13} - \theta_3 + \epsilon_{13}) \\
&= \phi_{12} + \phi_{23} - \phi_{13} + \epsilon_{12} + \epsilon_{23} - \epsilon_{13}.
\end{aligned} \tag{3.3}$$

Equation 3.3 clearly shows that the phase contribution of the gain terms to the closure phase (ψ_{123}) is zero. This it true for any closed baseline loop, regardless of the number of antennas.

There is an amplitude analogue for closure phase, known as the *closure amplitude* (Smith (1952), Twiss et al. (1960), though the term is due to Readhead et al. (1980)). It is computed by considering the amplitude ratios between baselines. Unlike closure phase, at least four elements/antennas are required to form a closure amplitude.

The closure amplitude for four antennas labelled 1 through 4 is given by:

$$\begin{aligned}
A_{1234} &= \frac{|v'_{12}||v'_{34}|}{|v'_{13}||v'_{24}|} \\
&= \frac{|g_1 v_{12} g_2||g_3 v_{34} g_4|}{|g_1 v_{13} g_3||g_2 v_{24} g_4|} \\
&= \frac{|v_{12}||v_{34}|}{|v_{13}||v_{24}|},
\end{aligned} \tag{3.4}$$

where all symbols retain their previous definitions and A_{1234} is a closure amplitude. As in the case of closure phase, the gain amplitudes do not appear in Equation 3.4.

Both of the closure quantities have been presented here because without them it is unlikely that self-calibration as it is today would exist. They are also interesting observables in their own right, although their interpretation can be challenging as they are derived from several visibility measurements. However, perhaps the most important point is that they can be used to recover information from the target field even when the gain terms are unknown. This notion of using the target itself to improve the observed data is the central idea behind self-calibration and its predecessors.

Rogers et al. (1974) developed one of the earliest of these predecessors, which made use of the closure phases and Fourier series to perform a model fitting procedure. This bears little resemblance to self-calibration as it stands today, but marks the beginning of a period of renewed interest in the problem. Ekers (1984) attributes this to the developments being made in the field of Very Long Baseline Interferometry (VLBI), where arrays were not phase-stable and the closure phases where necessary to access the true phase information.

It was the needs of the VLBI community that led to the development of *hybrid mapping* (term due to Baldwin and Warner (1978)) algorithms, self-calibration's closest antecedent. The "hybrid" of hybrid mapping refers to the process of making a map/image using a mixture of information from the observed visibilities and model (also referred to as trial) visibilities. The methods in question follow a series of steps (Pearson and Readhead 1984):

1. Obtain or construct a model map/image of the sky brightness distribution.
2. Generate model visibilities associated with the model map/image by Fourier transform.
3. Combine the observed visibilities and the model visibilities.
4. Produce a hybrid map/image by Fourier inversion of the combined visibilities.
5. Improve the model map/image by inspecting the hybrid map.

This is an iterative procedure, and the methods are considered to have converged when the hybrid map is indistinguishable from the model map/image. It is usually the particulars of steps 1 and 3 that distinguish the different methods.

The method of Fort and Yee (1976) is first of the hybrid mapping (though it predates the term) methods to follow these steps. The authors create a hybrid map (step 3 above) by using the phase information of the model visibilities and the amplitude information of the observed visibilities. At each iteration, the negative components in the hybrid map are set to zero and the resulting image is used as the model for subsequent iterations.

A second approach was devised by Baldwin and Warner (1978), though its principle goal was phaseless aperture synthesis. Like Fort and Yee (1976), the authors make use of phase information from the model visibilities while using the observed amplitudes. However, they show that an initial trial map can be obtained by inspecting an image of the two-dimensional autocorrelation function of the observed visibilities, i.e. the Fourier transform of the amplitude squared.

The approach taken by Readhead and Wilkinson (1978) brings us closer to the first self-calibration algorithm. Their method was inspired by the work of Fort and Yee (1976), with a few major differences. In order to produce a hybrid map, the authors set the phases of the observed visibilities on $N_{\mathrm{A}} - 1$ baselines equal to the phases of the corresponding model visibilities. The phases of the remaining baselines are then determined from the $(N_{\mathrm{A}}-1)(N_{\mathrm{A}}-1)/2$ closure phase relationships. The resulting hybrid map is subsequently deconvolved using the CLEAN algorithm (Högbom 1974), the output of which (a number of point sources) is used as the new model map/image on the next iteration. This approach was subsequently extended by Readhead et al. (1980) to make use of closure amplitudes.

Cotton (1979) developed a method similar to that of Readhead and Wilkinson (1978). Algorithmically, they differ in only one major respect; Cotton (1979) does not set $N_{\mathrm{A}}-1$ of the phases to those predicted by the model. Instead, he solves a constrained least-squares problem to correct the model visibility phases such that they agree with the closure phases. The author also draws attention to his use of a Direct Fourier Transform (DFT) as it circumvents the requirement that the visibilities be gridded for use with the Fast Fourier Transform (FFT).

It was only in 1980, with the work of Schwab (1980) and Cornwell and Wilkinson (1981), that the first true self-calibration methods emerged. Prior to their advances, hybrid mapping had always relied on closure quantities which were computed per-baseline. Schwab (1980) instead expressed the problem as a non-linear least squares (NLLS) fit between the data and a product of model visibilities with antenna based amplitude and phase errors (the antenna gains). He minimised the following:

$$S = \sum_{p \neq q} w_{pq} |d_{pq} - g_p m_{pq} \bar{g}_q|^2, \tag{3.5}$$

with respect to the gains. In this expression, w_{pq} is a weight associated with baseline pq, d_{pq} is the visibility observed on baseline pq, m_{pq} is the visibility predicted on baseline pq and g_p is the gain associated with antenna p. $|(\cdot)|$ denotes the absolute value and $\bar{(\cdot)}$ denotes conjugation.

Cornwell and Wilkinson (1981) extended Schwab's method to allow for different degrees of phase and amplitude instability at each antenna. The authors still described their own method as a hybrid mapping algorithm although the move from the explicit use of closure quantities to the determination of the antenna gains marks the transition from hybrid mapping to self-calibration.

3.2.2 The self-calibration loop

At this juncture, it seems appropriate to provide details of what has become the conventional self-calibration approach (see Figure 3.1). We will briefly describe each step in the process.

1. A (possibly initial) model is obtained. This may be an image of structure which is believed to be real or a list of source components. The manner in which the initial model is obtained varies. Perhaps the most common approach is to perform a shallow deconvolution, using something like CLEAN (Högbom 1974), and use the resulting CLEAN components as the initial model. It is also common to use source finding software such as PyBDSF (Mohan and Rafferty 2015) to extract a model from a restored image.

2. The model is converted into visibilities by means of the Fourier transform. This is referred to as the predict step as the output is the visibilities associated with the model. Many software packages have some form of prediction capability e.g. AIPS (Greisen 2016), CASA (McMullin et al. 2007), MeqTrees (Noordam and Smirnov 2010) and Montblanc (Perkins et al. 2015). The packages themselves often determine the kind of model necessary for step 1.

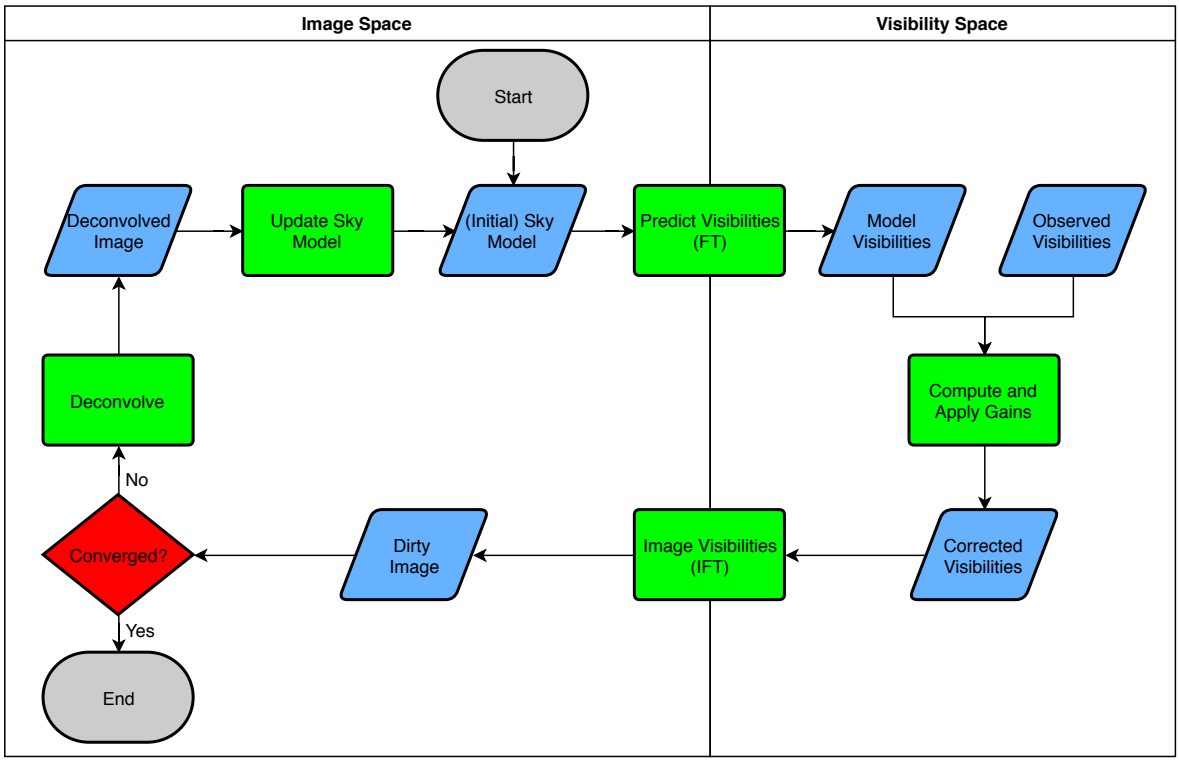

FIGURE 3.1: Flow chart describing the self-calibration process.

3. The model visibilities are used in conjunction with the observed visibilities to compute gains. This is the calibration step as the application of the inverse gains improves the observed data. The output is the corrected visibilities. There are a multitude of ways in which this step can be performed, some of which will be discussed later in this chapter.

4. A dirty image is formed by taking the inverse Fourier transform of the corrected visibilities. Several pieces of software can perform this imaging, but the most ubiquitous are AIPS (Greisen 2016), CASA (McMullin et al. 2007) and WSCLEAN (Offringa et al. 2014).

5. If the image obtained in step 4 is considered good enough, there is no need to go further (beyond some post-processing and a final round of deconvolution) and the process is considered to have converged. The notion of good enough is completely subjective, but it is usually at the point that the calibration artefacts/problems are no longer obvious in the image. If the image is not yet satisfactory, it is necessary to continue with the self-calibration loop.

6. The dirty image obtained in step 4 is deconvolved to produce a restored image which is used, in turn, to update the sky model. The form of this update depends on the format of the model e.g. a better model image or a more complete/longer list of source components. This updated model is then used as the input to step 1 and the entire process is repeated until step 5 is satisfied.

3.2.3 2GC Methods

Having concluded the discussion of the origins and basics of generic self-calibration, it makes sense to describe some of the different approaches to step 3 of the self-calibration loop. We will endeavour to provide some insight into how they work, though we will omit the more involved derivations.

Thompson and D'Addario's method

One of the first methods for determining the antenna gains was proposed by Thompson and D'Addario (1982) and a variation of the method still appears in one of the most widely used references in the field of interferometry (Thompson et al. 2017). For the purposes of this discussion we will follow Bhatnagar and Nityananda (2001), which presents a more satisfactory derivation of the method. The function which is to be minimised is identical to that proposed by Schwab (1980) (see Equation 3.5). However, the first step is a division of the right hand side of the expression by the model visibilities, m_{pq}, yielding:

$$S = \sum_{p \neq q} w_{pq} |x_{pq} - g_p \bar{g}_q|^2, \tag{3.6}$$

where x_{pq} is the ratio of the observed and model visibilities. This expression can be expanded, noting that for a complex number z, $|z|^2 = z\bar{z}$. Consequently, Equation 3.6 becomes:

$$S = \sum_{p \neq q} w_{pq} \left(x_{pq} - g_p \bar{g}_q\right) \left(\bar{x}_{pq} - \bar{g}_p g_q\right). \tag{3.7}$$

In order to go further using conventional differentiation, it is necessary to split the components of Equation 3.7 into their real and imaginary parts. This is to avoid treating $\partial \bar{g} / \partial g$, which does not exist. Making this split and grouping the terms gives:

$$S = \sum_{p \neq q} w_{pq} Z_{pq} \bar{Z}_{pq}, \tag{3.8}$$

where:

$$Z_{pq} = \left(x^R_{pq} - g^R_p g^R_q - g^I_p g^I_q\right) + i \left(x^I_{pq} + g^R_p g^I_q - g^I_p g^R_q\right). \tag{3.9}$$

The superscripts R and I denote the real and imaginary components of the quantities with which they are associated. Differentiating Equation 3.8 with respect to the real component of the a'th gain, g^R_a, we obtain (after a slew of tedious simplification):

$$\frac{\partial S}{\partial g^R_a} = -2 \sum_{q, q \neq a} w_{aq} \left[\left(x_{aq} g_q\right)^R - g^R_a |g_q|^2\right]. \tag{3.10}$$

Setting this expression to zero and rearranging the terms yields an update rule for the real component of the a'th gain:

$$g_a^R = \frac{\sum\limits_{q,q\neq a} w_{aq}(x_{aq}g_q)^R}{\sum\limits_{q,q\neq a} w_{aq}|g_q|^2}. \tag{3.11}$$

The update rule for the imaginary component can be derived in a similar fashion, producing:

$$g_a^I = \frac{\sum\limits_{q,q\neq a} w_{aq}(x_{aq}g_q)^I}{\sum\limits_{q,q\neq a} w_{aq}|g_q|^2}. \tag{3.12}$$

Noting that the overall update to the gain is given by $g_a = g_a^R + ig_a^I$, we combine Equations 3.11 and 3.12 and obtain:

$$g_a = \frac{\sum\limits_{q,q\neq a} w_{aq}x_{aq}g_q}{\sum\limits_{q,q\neq a} w_{aq}|g_q|^2}, \tag{3.13}$$

which can be applied iteratively until an acceptable gain solution is reached. This approach seems to have been the norm for quite some time. However, it is only suitable for treating a single correlation at a time; it does not generalise to the case of multi-correlation (and therefore polarisation) data.

Hamaker's method

Hamaker (2000) began to address the difficulties of calibrating four-correlation data by reformulating the problem in terms of 2×2 Jones matrices. However, it should be noted that the derivation he employed is not particularly intuitive. Consequently, we will approach the problem slightly differently but will ultimately obtain an identical result.

We begin from the same point as Hamaker (ibid.); expressing the function which we want to minimise as:

$$S = \sum_{pq} \left\| \mathbf{G}_p^{-1}\mathbf{D}_{pq}\mathbf{G}_q^{-H} - \mathbf{M}_{pq} \right\|_F^2, \tag{3.14}$$

the elements of the which are all 2×2 matrices. $\mathbf{D}_{pq}$ contains the data (up to four correlations) observed on baseline pq, $\mathbf{M}_{pq}$ is the model visibility which is expected on baseline pq and $\mathbf{G}_p^{-1}$ is the inverse of the gain associated with antenna p. $(\cdot)^{-1}$ denotes a matrix inverse, $(\cdot)^H$ denotes a conjugate transpose and $(\cdot)^{-H}$ is a combination of both operations. $\|\cdot\|_F^2$ is the Frobenius norm squared which was referred to as the variance by Hamaker (ibid.). The goal is to solve for the inverse of the antenna gains, $\mathbf{G}_p^{-1}$. One peculiarity of framing the problem is this fashion is that it describes the process of fitting noisy data to a model as opposed to the more conventional (and strictly speaking correct) approach of fitting a model to the noisy

data as described by:

$$S = \sum_{pq} \left\| \mathbf{D}_{pq} - \mathbf{G}_p \mathbf{M}_{pq} \mathbf{G}_q^H \right\|_F^2. \tag{3.15}$$

In the original work, Hamaker (ibid.) approached the problem by considering the differentials of the objective function S. We will present an alternative derivation which we believe is more intuitive and connects more clearly to subsequent work in the field. To do so, we will consider the following matrix equation:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3.16}$$

where $\mathbf{A}$ is an $N \times M$ matrix, $\mathbf{x}$ is an $M \times 1$ vector of parameters and $\mathbf{b}$ is an $N \times 1$ vector of data points. If a system of equations can be written in this form, then a solution may be found using the *normal equations* which for complex-valued problems are given by:

$$\mathbf{A}^H \mathbf{A}\mathbf{x} = \mathbf{A}^H \mathbf{b}. \tag{3.17}$$

Before we can make use of Equation 3.17, it is first necessary to express Equation 3.14 in a vector form compatible with Equation 3.16. In order to do this, we rewrite equation 3.14 as:

$$S = \sum_{pq} \left\| \mathrm{vec}(\mathbf{G}_p^{-1} \mathbf{D}_{pq} \mathbf{G}_q^{-H}) - \mathrm{vec}(\mathbf{M}_{pq}) \right\|_2^2. \tag{3.18}$$

We then make use of the following useful property:

$$\mathrm{vec}(\mathbf{LXR}) = (\mathbf{R}^T \otimes \mathbf{L})\,\mathrm{vec}(\mathbf{X}), \tag{3.19}$$

noting that because $\mathbf{G}_p^{-1}$ is the parameter of interest, we choose $\mathbf{L} = \mathbf{I}$, $\mathbf{X} = \mathbf{G}_p^{-1}$ and $\mathbf{R} = \mathbf{D}_{pq}\mathbf{G}_q^{-H}$ and write Equation 3.18 as:

$$S = \sum_{pq} \left\| ((\mathbf{D}_{pq} \mathbf{G}_q^{-H})^T \otimes \mathbf{I})\,\mathrm{vec}(\mathbf{G}_p^{-1}) - \mathrm{vec}(\mathbf{M}_{pq}) \right\|_2^2. \tag{3.20}$$

All that is required now is to define the matrices which give the system of $2N_{\mathrm{BL}}$ equations (each baseline pq and their conjugate) associated with Equation 3.20. Returning to Equation 3.16, we find that the entries of $\mathbf{A}$ (each is in fact a 4×4 matrix) are given by:

$$\mathbf{A}_{a \to pq, b} = ((\mathbf{D}_{pq} \mathbf{G}_q^{-H})^T \otimes \mathbf{I}) \delta_p^b, \tag{3.21}$$

where the rows and columns of $\mathbf{A}$ are indexed by a and b respectively and δ is the Dirac delta ($\delta_p^b = 1$ if $p = b$ else 0). The somewhat unusual subscript $a \to pq$ is used to clarify the fact that each row in $\mathbf{A}$ is associated with a specific baseline. By convention, this baseline index is chosen to enumerate all baselines ($p < q$) and their conjugates ($p > q$). Additionally, the entries of $\mathbf{x}$ and $\mathbf{b}$ are given by:

$$\mathbf{x}_a = \mathrm{vec}(\mathbf{G}_a^{-1}), \qquad \text{and} \qquad \mathbf{b}_{a \to pq} = \mathrm{vec}(\mathbf{M}_{pq}). \tag{3.22}$$

The vector $\mathbf{x}$ has N_{A} entries and $\mathbf{b}$ has $2N_{\mathrm{BL}}$ entries. Note that the entries of both of these vectors are vectorised 2×2 matrices.

We are finally in a position to use the normal equations to write down an update rule. First, we note that the entries of $\mathbf{A}^H$ are given by:

$$\mathbf{A}^H_{a,b\to pq} = ((\mathbf{D}_{pq}\mathbf{G}_q^{-H})^* \otimes \mathbf{I})\delta^a_p, \tag{3.23}$$

where a indexes the rows and b indexes the columns, and $(\cdot)^*$ denotes conjugation. Due to the transposition, it is the columns of $\mathbf{A}^H$ that are associated with specific baselines. Using these expressions for $\mathbf{A}$ and $\mathbf{A}^H$, we can write the entries of their product as:

$$(\mathbf{A}^H\mathbf{A})_{ab} = \begin{cases} \sum_{q\neq a}((\mathbf{D}_{aq}\mathbf{G}_q^{-H})^*(\mathbf{D}_{aq}\mathbf{G}_q^{-H})^T \otimes \mathbf{I}), & \text{for } a = b \\ 0, & \text{for } a \neq b \end{cases}. \tag{3.24}$$

Similarly, we can obtain an expression for the entries of $\mathbf{A}^H\mathbf{b}$:

$$(\mathbf{A}^H\mathbf{b})_a = \sum_{q\neq a}((\mathbf{D}_{aq}\mathbf{G}_q^{-H})^* \otimes \mathbf{I})\,\mathrm{vec}(\mathbf{M}_{aq}). \tag{3.25}$$

Finally, the product of $\mathbf{A}^H\mathbf{A}$ with $\mathbf{x}$ produces a vector, the entries of which are given by:

$$(\mathbf{A}^H\mathbf{A}\mathbf{x})_a = \sum_{q\neq a}((\mathbf{D}_{aq}\mathbf{G}_q^{-H})^*(\mathbf{D}_{aq}\mathbf{G}_q^{-H})^T \otimes \mathbf{I})\,\mathrm{vec}(\mathbf{G}_a^{-1}) \tag{3.26}$$

Substitution of Equations 3.25 and 3.26 into 3.17 already provides a method of solving for $\mathrm{vec}(\mathbf{G}_a^{-1})$. However, by reversing the trick used in 3.19, we obtain an identical result to Hamaker (2000):

$$\mathbf{G}_a^{-1}\sum_{q\neq a}\mathbf{D}_{aq}\mathbf{G}_q^{-H}\mathbf{G}_q^{-1}\mathbf{D}_{aq}^H = \sum_{q\neq a}\mathbf{M}_{aq}\mathbf{G}_q^{-1}\mathbf{D}_{aq}^H. \tag{3.27}$$

For the sake of consistency, we rearrange Equation 3.27 and obtain:

$$\mathbf{G}_a^{-1} = \left(\sum_{q\neq a}\mathbf{M}_{aq}\mathbf{G}_q^{-1}\mathbf{D}_{aq}^H\right)\left(\sum_{q\neq a}\mathbf{D}_{aq}\mathbf{G}_q^{-H}\mathbf{G}_q^{-1}\mathbf{D}_{aq}^H\right)^{-1}, \tag{3.28}$$

which can be applied iteratively to find an estimate of the a'th gain's inverse, $\mathbf{G}_a^{-1}$.

Equation 3.28 is the same as that used by Mitchell et al. (2008) in the real time calibration system of the MWA (Murchison Widefield Array). What is peculiar is that Mitchell et al. (ibid.) also end up fitting the data to the model even though initially they express the more conventional case of fitting the model to the data (see Equation 3.15). The given reason for the change was that they found that the solutions for the case of Equation 3.15 were unstable.

Another interesting result which comes out of this derivation is that the method proposed by Thompson and D'Addario (1982) can also be derived by applying the normal equations to Equation 3.6.

StEFCal

Salvini and Wijnholds (2014) developed StEFCal (Statistically Efficient and Fast Calibration), as a means of calibrating the current generation of radio interferometers, which have a far larger number of receivers than older instruments. The authors describe their method as an alternating direction implicit (ADI) approach, but in truth this just encapsulates the assumption that the gains can be determined whilst treating their conjugates as constant.

Under this assumption, it is possible to use the normal equations (as was done in our derivation of Hamaker's approach) to determine an update rule. One crucial difference between StEFCal and Hamaker's update rule is the objective function. Salvini and Wijnholds (ibid.) use the following (identical to Equation 3.15):

$$S = \sum_{pq} \left\| \mathbf{D}_{pq} - \mathbf{G}_p \mathbf{M}_{pq} \mathbf{G}_q^H \right\|_F^2, \tag{3.29}$$

where all symbols retain their previous definitions. This choice of objective function yields the following per-antenna update rule:

$$\mathbf{G}_a = \left(\sum_{q \neq a} \mathbf{D}_{aq} \mathbf{G}_q \mathbf{M}_{aq}^H \right) \left(\sum_{q \neq a} \mathbf{M}_{aq} \mathbf{G}_q^H \mathbf{G}_q \mathbf{M}_{aq}^H \right)^{-1}. \tag{3.30}$$

Salvini and Wijnholds (ibid.) do not go through the derivation of the four-correlation form, and instead only consider the scalar (single correlation) case. We have chosen to present the full polarisation version for the sake of consistency and to make it clear how similar the resulting update rule is to that derived by Hamaker (2000). Additionally, we have now shown that all the major 2GC algorithms can be obtained from the normal equations.

Salvini and Wijnholds (2014) went a step further and made a study of the convergence behaviour of their algorithm, noting that convergence behaviour could be substantially improved by averaging every even iteration with the preceding odd iteration. That is, for iteration index k, setting:

$$\mathbf{G}_a^k = (\mathbf{G}_a^k + \mathbf{G}_a^{k-1})/2 \quad \text{if } k \text{ is even.} \tag{3.31}$$

In truth, this averaging step is effectively the manifestation of a conventional damping parameter (λ) which tunes the step size between iterations. Consequently, there is no guarantee that averaging every even iteration is optimal. However, it has been shown to work well empirically, and avoids the additional computation required to compute a truly optimal value for λ.

One final contribution made by Salvini and Wijnholds (ibid.) is that the update rule can be written entirely in terms of large matrices. We will not describe this in detail here, as it forms part of a later explanation, but it is important to note that this can be exploited from a computational perspective.

3.3 Third Generation Calibration (3GC)

Third generation calibration refers to the treatment of direction-dependent effects (DDEs). These are errors which cannot be adequately modelled by a direction-independent (2GC) gain as they vary across the field of view. This is an area of active research, as the performance of modern instruments may be severely limited by DDEs. Before briefly discussing some of the approaches to direction-dependent gain calibration, it is important to establish the most common (and severe) sources of these errors.

Smirnov (2011b) provides a detailed review of many of the sources of DDEs. Therein, a distinction is made between trivial DDEs (DDEs which are treated as being the same for all antennas and times) and non-trivial DDEs. Non-trivial DDEs are the result of each baseline effectively seeing a different sky, and require special treatment. What follows are some of the most troublesome and well-known DDEs, as noted by Smirnov (ibid.).

3.3.1 Primary beam effects

The primary beam is an omnipresent source of DDEs. All telescopes have a some form of beam, and it varies from one array to another. It is so ubiquitous in fact, that the symbol used to represent the primary beam within the RIME, $\mathbf{E}$, is also used to refer to generic direction-dependent Jones terms.

The primary beam has often been treated as a trivial DDE which is removed by a multiplication of the output images with the inverse of the power beam. This corrects the flux scale, but inflates the noise in regions of low beam gain. In the context of the RIME, this approach is equivalent to having a scalar $\mathbf{E}$ term. As the beam is assumed to be the same across all antennas when treated as a trivial DDE, any phase errors it would introduce cancel out.

However, the effect of the primary beam is, in fact, far more complicated and the conditions for a trivial DDE (being the same for all times and all antennas) are definitely unmet. Each antenna will almost certainly have a slightly different beam. These differences can be attributed to a host of factors, from each dish simply being physically non-identical to differences in environmental conditions (which of course vary with time).

An antenna's receptors can also have differing beam patterns. In the simplest case, this manifests as a diagonal $\mathbf{E}$ term which introduces instrumental polarisation. That is, the beam can cause an unpolarised source to appear polarised.

One of the more obvious effects, which further complicates the problem, is pointing errors. Pointing errors are small deviations (of the order of a small fraction of the beam size) from the expected antenna pointing. Even if every antenna had an identical beam, the pointing errors would be sufficient to cause a non-trivial DDE as each source would be in a slightly different position in each antenna's beam.

The type of telescope also plays a part in determining the extent to which the primary beam affects observations. The beam pattern of an alt-az mount telescope rotates in the lm-plane due to parallactic angle rotation. This implicitly makes the beam patterns time-dependent.

3.3.2 Atmospheric effects

The atmosphere introduces direction-dependent phase errors, particularly at low frequencies. The main cause of these phase errors is the ionosphere which introduces a phase delay as a result of refraction. The extent to which it effects an observation depends on several parameters, including the field of view, size of the array and scale on which the ionosphere varies. Smirnov (ibid.), following Lonsdale (2005), uses these to describe observational regimes and establish the extent to which the ionosphere is expected to affect them. We will not reproduce the regimes here for the sake of brevity, but suffice it to say that the ionosphere is a non-trivial DDE for many contemporary and future instruments.

A secondary effect of the ionosphere is Faraday rotation (a rotation of the plane of polarisation). This effect is proportional to the inverse of frequency squared (ν^{-2}) and is consequently negligible at high frequencies. It can, however, have a large effect for low frequency arrays. In particular, the effect of differential Faraday rotation (DFR) can be quite dramatic on long baselines. This is because the degree of rotation at each antenna can differ by quite a large amount. In the most extreme case, DFR can cause all the flux of an unpolarised source to be detected as polarised emission.

Finally, it is worth mentioning that the troposphere also introduces a phase delay in a similar fashion to the ionosphere. However, as the troposphere is at lower altitude, it can usually be treated as a direction-independent effect. Practically, it will usually be absorbed into the conventional 2GC gain solutions.

3.3.3 3GC Methods

The requirements of contemporary and future instruments have caused a proliferation of direction-dependent calibration algorithms/strategies. Some of these are extensions of conventional self-calibration, whilst others approach the problem in new and sometimes peculiar ways.

We will concentrate on describing some of the better understood and mature approaches. The alternatives will be discussed briefly, but they often involve branches of mathematics well beyond the scope of this document.

Peeling

One of the earliest methods of solving for direction-dependent gain solutions, the so-called *peeling* algorithm (Noordam 2004), emerged from the requirements of the LOFAR (Low Frequency Array) in the Netherlands. Atmospheric effects are particularly problematic in LOFAR's observing bands, and vary widely across the field-of-view which itself may be very large.

Conventional self-calibration solutions are dominated by the brightest sources in the field. Consequently, self-calibration corrections are best in the vicinity of those sources. However, in the presence of DDEs (such as those introduced by the atmosphere) many of the fainter sources will remain poorly calibrated even after the application of direction-independent gain solutions. Peeling is a intuitive extension to self-calibration that addresses this problem by

calibrating sources individually in order of descending brightness. The following is a simple peeling strategy:

1. Select the (next) brightest source.
2. Determine self-calibration solutions for the selected source.
3. Subtract/peel the contribution of the selected source, multiplied by the relevant gains, from the visibilities.
4. Return to step one and repeat using the updated visibilities.

In principle, peeling can be performed for any number of sources (N_S), but rapidly becomes prohibitively expensive as it requires solving N_S self-calibration problems in series. Additionally, each source which is peeled introduces additional free parameters (usually N_A direction-dependent gains) and overfitting becomes a genuine concern. Peeling is thus only suited to solving in the direction of the very brightest sources.

Smirnov (2011b) also notes that peeling "freezes" in any errors at each iteration. This can be mitigated by performing the peeling procedure several times, but further increases the computational expense.

Some of the errors which appear while peeling stem from a problem which plagues calibration in general - the use of an incomplete model. This leads to the appearance of calibration artefacts known as *ghosts* (Grobler et al. 2014, 2016; Wijnholds et al. 2016). Selecting only the brightest source is equivalent to using an incomplete model and care must be taken to ensure that the results are not contaminated by spurious ghost sources. Noordam (2004) does remark that some of the errors can be mitigated by including additional sources in the model whilst only solving for the gains in the direction of the brightest source. Once again, this comes with a commensurate increase in computational cost.

However, even taking all of its problems into account, peeling is still the most well-understood and most widely implemented technique of solving for direction-dependent gains. This is, in part, due to the fact that it can be implemented as a simple extension to existing self-calibration implementations.

Pointing self-cal

The problems introduced by the primary beam have already been discussed in Subsection 3.3.1. Given their prevalence, it is natural that some attempt has been made to correct for them during calibration. *Pointing self-cal* (Bhatnagar and Cornwell 2017; Bhatnagar et al. 2004) attempts to solve for the per-antenna pointing errors. The resulting errors can be incorporated into the *A-projection* algorithm (Bhatnagar et al. 2008, 2013) which takes the beam into account during imaging.

The pointing self-cal algorithm itself follows the same basic structure as traditional self-calibration, with iterative improvements to both the model and the parameters. However, pointing self-cal is an aperture-plane algorithm and the effect of the primary beam is applied directly to the visibilities. This differs from the more conventional approach of solving for

the direction-dependent gains in image space where it is necessary to solve in the direction of several sources.

In order to perform calibration in the aperture plane, pointing self-cal considers the convolution of the Fourier transform of the antenna beams with the model visibilities. The antenna beams themselves are parameterised by the per-antenna pointing errors. This parameterisation requires either an analytical expression for the beam, or a sufficiently accurate model from which numerical derivatives can be obtained.

Bhatnagar and Cornwell (2017) make the simplifying assumption that the primary lobe of the beam patterns can be described by a Gaussian. Using this assumption, they obtain analytic expressions for the derivative of the Fourier transform of the beams with respect to the pointing errors. Finally, these expressions are used in conjunction with A-projection to determine a parameter update for the pointing errors. The form of this update depends on the non-linear minimisation method which is employed.

The SAGE algorithm

The SAGE (Space Alternating Generalised Expectation-maximisation, Fessler and Hero (1994)) algorithm was first applied to the radio interferometric calibration problem by Yatawatta et al. (2008) as an alternative to the ubiquitous least-squares methods. It is an extension of the EM (Expectation-Maximisation, Dempster et al. (1977)) algorithm. The SAGE method was said to achieve higher accuracy than its competitors at a lower computational cost. This is of the utmost importance in light of the size and sensitivity of instruments like LOFAR and the SKA.

Kazemi et al. (2011) provides a more complete description of the application of the SAGE algorithm to radio interferometry and it is to this paper that we will refer for a description of the method.

The central idea behind EM and SAGE is to augment the observed/incomplete data with a so-called complete-data space for which estimation is tractable. This leads to a two-step iterative solution scheme comprised of an expectation step and a maximisation step. The expectation step involves calculating the conditional expectation of the complete-data log-likelihood. The maximisation step computes a parameter update by performing some sort of maximisation on the expression found during the expectation step. Fessler and Hero (1994) states that the principle difference between the EM and SAGE algorithms is that EM attempts to update all parameters simultaneously while SAGE updates only a subset of the parameters per iteration by splitting the complete-data space into several pieces (hidden-data spaces). Each hidden-data space is associated with a subset of the parameters.

For the specific problem of radio interferometric calibration these steps can be simplified to give an intuitive picture of what the SAGE algorithm does. The expectation step yields the following expression:

$$\hat{\mathbf{x}}_i^k = \mathbf{d} - \sum_{s \neq i}^{N_S} \mathbf{m}_s(\mathbf{e}_s^k), \tag{3.32}$$

where $\mathbf{x}$ is the hidden-data space, $\hat{\mathbf{x}}$ is its conditional expectation value, $\mathbf{d}$ is the observed data and $\mathbf{m}$ are model visibilities which depend on the direction dependent gains $\mathbf{e}$. The superscript k denotes iteration, subscript i indicates the source for which the parameters are to be updated and the sum is over N_{S}; the number of sources in the model. These vectors all have $8N_{\mathrm{BL}}$ entries - a factor of four due to vectorising 2×2 Jones terms and a factor of two due to splitting the real and imaginary parts.

Equation 3.32 may seem complicated but is, in fact, very similar to peeling. $\hat{\mathbf{x}}_i$ is simply the current best estimate of the visibilities for source i or, in other words, the data minus the contribution of the $N_{\mathrm{S}} - 1$ other sources in the model and their associated gains. This is expected to become more accurate as the values of $\mathbf{e}_s$ are updated. Note that the sum could also be taken over groups/clusters of sources.

The maximisation step then involves minimising a cost function of the form:

$$S(\mathbf{e}_i) = \left\| \hat{\mathbf{x}}_i^k - \mathbf{m}_i(\mathbf{e}_i^k) \right\|^2, \tag{3.33}$$

or a weighted version thereof. All symbols retain their previous definitions. Following on from the discussion of Equation 3.32, we can see that Equation 3.33 describes a fit between $\hat{\mathbf{x}}_i$ (the data minus the contributions of all sources other than source i) and the model for source i. The cost function S is usually minimised using a least-squares method, in a similar fashion to conventional calibration. This yields an updated estimate of $\mathbf{e}_i$. Each source (i) is updated sequentially and the SAGE algorithm ultimately converges to a maximum-likelihood solution for all sources.

The SAGE algorithm was extended to make use of ordered subsets (OS, Hudson and Larkin (1994)) by Kazemi et al. (2013). The OS-SAGE algorithm makes use of partitioned data i.e. the input observed data is grouped into sub-observations, and the algorithm iterates over the sub-observations to refine the parameter estimates. The advantage of this approach is that it reduces the data volume per SAGE iteration. Note that the improvement is one of speed/computational complexity and OS-SAGE is at most only as accurate as the SAGE algorithm.

Robust calibration

Robust calibration (Kazemi and Yatawatta 2013; Yatawatta and Kazemi 2014) was developed to overcome one of the most prevalent problems in radio interferometry; outliers in the data. Kazemi and Yatawatta (2013) attribute these outliers to several causes, most notably RFI (Radio Frequency Interference) and the use of incomplete/incorrect models during calibration.

The presence of these outliers violates one of the fundamental assumptions of calibration; that the noise is Gaussian. This can cause traditional calibration methods to perform sub-optimally and introduce artefacts/ghosts (Grobler et al. 2014, 2016; Wijnholds et al. 2016). These artefacts can both suppress real flux and cause the appearance of spurious emission. Consequently, Kazemi and Yatawatta (2013) define robustness - in the context of their method - as the degree to which the fluxes of unmodelled sources are preserved.

Conventional least-squares minimisation on its own is insufficient to treat the problem of non-Gaussian noise. As such, Kazemi and Yatawatta (ibid.) extend their previous work (Kazemi et al. 2011; Yatawatta et al. 2008) to frame calibration as an iteratively reweighted least-squares problem. This is done using the ECME (Expectation/Conditional Maximisation Either, Liu and Rubin (1995)) algorithm, which is itself an extension of the EM algorithm (Dempster et al. 1977).

Many of the details of robust calibration are identical to the SAGE algorithm. One important difference is that the problem is framed in terms of the effective noise, i.e. the true Gaussian noise plus a non-Gaussian component introduced by the unmodelled sources. This is incorporated by treating the effective noise as following a Student's t-distribution. The t-distribution is similar to a Gaussian distribution but with heavier tails. It is these tails that capture the behaviour of the outliers.

The Student's t-distribution is incorporated into the ECME algorithm and is used to iteratively reweigh the data. The basic operation of the algorithm can be summarised as follows:

1. Given initial values for the unknown parameters and weights, use conventional least-squares to update the unknowns (gains) k times or until a stopping criterion is reached. This is essentially a maximisation step in the context of EM.

2. Using the updated values of the unknowns, update the weights. In practice, this involves several (complicated) steps but boils down to finding new estimates of the weights given the current values of the unknowns.

3. Carry out additional maximisation steps (see step 1) using the new weights.

4. Repeat steps 2 and 3 l times. l is the maximum number of EM iterations chosen.

Ollier et al. (2016, 2017) have extended robust calibration by modelling the effective noise as a spherically invariant random distribution. The Student's t-distribution is just one such distribution. Consequently, the method of Ollier et al. (2017) makes fewer assumptions about the underlying behaviour of the effective noise and should be more robust than its predecessors. Ollier et al. (ibid.) make use of conventional EM to solve for the unknown parameters, but augment the maximisation step with a BCD (Block Coordinate Descent) algorithm (Friedman et al. 2007) to accelerate computation.

Compressed sensing

The methods of compressed sensing (Candes et al. 2005; Donoho 2006) have already been applied successfully to the radio interferometric imaging problem (see, for example, Repetti et al. (2017) and the references therein). Whilst a full description of the framework is beyond the scope of this work, we will note in passing that compressed sensing exploits the sparsity of a signal in some domain to perform signal reconstruction.

These methods have been applied directly to the calibration problem by Kazemi et al. (2015). They consider the problem of *blind calibration* - calibration in the absence of a

sky model. The earliest form of blind calibration is *redundant calibration* (Noordam and de Bruyn 1982) which exploits redundant baselines (baselines sampling the same UV points) to simultaneously solve for both the gains and the model visibilities. Many modern instruments lack the redundancy to exploit this technique.

The blind calibration method of Kazemi et al. (2015) considers the sparsity of the signal in the voltage/signal domain (pre-correlation). At low SNR, which is often the case for a single integration, only the strongest signals will be above the noise. Consequently, the signal is effectively sparse and the values of the unknown gains and model are estimated using convex optimisation. This has been shown to be effective on LOFAR data.

Wijnholds et al. (2016) and Chiarucci and Wijnholds (2018) explore a different blind calibration method wherein both the model and the gains are determined in an iterative fashion. This is a similar strategy to conventional self-calibration, but coalesced into a single algorithm. An active set method (Mouri Sardarabadi et al. 2016) is used to estimate the source powers (model image) whilst keeping the gains fixed. The gains are then solved for using StEFCal and the estimated source powers. These two steps are repeated until a convergence criterion is reached. The efficacy of this approach has been demonstrated on LOFAR data. Note that the sparsity in this approach is in the image domain.

It would be remiss of us not to note that the blind calibration techniques of Kazemi et al. (2015) and Chiarucci and Wijnholds (2018) do not solve directly for direction-dependent gains. However, they are presented here because they are on the bleeding edge of calibration techniques and both incorporate effects such as the beam in their derivations.

Chiarucci and Wijnholds (ibid.) is just one example of compressed sensing techniques blurring what has, heretofore, been a relatively solid delineation between calibration and imaging techniques. Another example of this is found in Repetti et al. (2017). The authors use non-convex optimisation to perform direction-dependent self-calibration and imaging. They do this using a block-coordinate forward-backward approach; at each iteration they find an approximate solution for the direction-dependent gains before re-estimating an approximate image. The approximation here refers to the fact that each step is performed a set number of times, rather than until convergence. This iterative approach is of course similar to conventional self-calibration, though once again here it is coalesced in to a single algorithm/implementation with convergence guarantees.

Consensus optimisation

In the era of the SKA and its ilk, more and more attention is being paid not only to the quality of calibration, but to its computational feasibility. In fact, the only environment in which calibrating the SKA seems possible is a distributed one i.e. multiple compute nodes connected via a network.

Yatawatta (2015) has already developed one such distributed calibration scheme which recasts the calibration problem as a distributed optimisation problem which can be solved using consensus optimisation (Boyd et al. 2011). Specifically, the author treats the case where each compute agent only has access to a fraction of the entire bandwidth but where smoothness of the calibration parameters with frequency is required. This is accomplished

by using the alternating direction method of multipliers (ADMM) adapted for consensus optimisation (C-ADMM, Boyd et al. (ibid.)).

In this framework, each compute agent performs calibration on its locally available data (using a Riemannian trust-region algorithm, see Yatawatta (2013)) before passing its local updates to a single, central node called the fusion centre. It is at the fusion centre that the local updates are combined. The fusion centre ultimately returns values to the nodes that ensure smoothness across frequency.

This distributed approach has been further extended by Yatawatta et al. (2017), in which the authors consider modifications to some of the parameters associated with the ADMM algorithm and finds that convergence can be improved. Additionally, Yatawatta et al. (2018) goes on to consider the case where there is more data than available compute nodes and implements a data multiplexing scheme to overcome this limitation.

Other approaches

The list of methods presented thus far is not exhaustive but does feature most of the well-known approaches. Before moving on, however, there are more approaches which are worth mentioning but do not fit neatly into the previous sections.

Field based calibration (Cotton 2007) is a method of correcting for phase errors introduced by the ionosphere. These phase errors manifest as positional shifts and distortions which vary across the field of view. Field based calibration attempts to solve this problem by making several snapshot images, deconvolving them and determining the position offsets of the brightest sources (relative to their known true positions) in each snapshot. This yields a number of time-dependent position offsets which are used to fit a time-variable phase screen across the entire field of view. Low order Zernicke polynomials are used for this purpose. Finally, the phase screen can be incorporated in imaging and deconvolution (using faceting) to correct for the phase errors.

Intema et al. (2009) has developed SPAM (Source Peeling and Atmospheric Modelling), a direction-dependent calibration technique which is tailored to the problem of ionospheric calibration at low frequencies. It can be considered an extension of peeling (Noordam 2004) - the phase solutions found via peeling in the directions of the brightest sources are used to fit a phase screen (a linear combination of Karhunen-Loève basis functions) across the field of view. The phase screen then allows for the interpolation of ionospheric distortions to unsampled (unpeeled) directions, which can, in turn, be used to correct the phases per-facet during facet imaging.

More recently, Enßlin et al. (2014) has approached the calibration problem using information field theory and Bayesian statistics. This approach is far more rigorous than more conventional self-calibration, but is considerably more complicated from a mathematical perspective. The authors note that without proper treatment of the uncertainties of the calibration signal, there is a systematic bias in the solutions. They also suggest that using a non-parametric, signal-to-noise filtered calibration would be superior to the fairly standard use of solution intervals which are at best a piece-wise approximation of the true gains.

Finally, the application of complex optimisation to radio interferometric gain calibration (Smirnov and Tasse 2015) has not been discussed. The reason for this omission is simple - it is crucial to this work and will be discussed in greater detail in subsequent chapters. It is, however, important to note that what follows is neither the first nor only work in the field. Both Tasse (2014b) and Smirnov and Tasse (2015) suggest algorithms for gain calibration, some of which have been implemented (see Section 3.4).

3.4 Calibration software

Having discussed a variety of calibration algorithms, it is of course important to make mention of the various pieces of software that implement them. Whilst an effort will be made to associate specific software suites with algorithms, that information is often unavailable and some packages treat calibration as a black box.

AIPS

The AIPS[1] (Astronomical Image Processing System) package was developed at the NRAO (National Radio Astronomy Observatory), beginning in 1978 (Greisen 2016; Greisen and Bridle 1985). Coded in a mix of of Fortran, C and shell script, its main purpose was the calibration and imaging of radio interferometric data. It became the primary tool for these purposes at the VLA (Very Large Array) in 1981.

In 1983, AIPS was chosen to handle the data reduction of the VLBA (Very Long Baseline Array), a VLBI (Very Long Baseline Interferometry) instrument which was under construction. Some of its VLBI calibration routines, particularly global fringe fitting, remain in use today.

The longevity of the AIPS package does much to recommend it, even though it is slowly becoming obsolete as modern alternatives are developed. In fact, in 1992 it was resolved by the NRAO and other institutions that AIPS would be replaced with AIPS++ (now called CASA), which would feature modern programming techniques and languages. As a result, development on AIPS has slowed substantially but the package is still maintained and used, primarily by the VLBI community.

Calibration (specifically of the per-antenna complex gains) in AIPS is performed using a least-squares solution based on the work of Thompson and D'Addario (1982). However, it also has the functionality to perform the optimisation using an L1-norm (sum of the absolute values of the residuals) which is less sensitive to outliers in the data.

GIPSY

GIPSY[2] (Groningen Image Processing System, van der Hulst et al. (1992)) was originally developed in 1971 for processing WSRT continuum data. Over the course of many years, it saw numerous upgrades and improvements and became applicable to instruments other than the WSRT, particularly IRAS (Infrared Astronomical Satellite).

[1] http://www.aips.nrao.edu/index.shtml
[2] https://www.astro.rug.nl/~gipsy/index.html

GIPSY was implemented primarily in Fortran and now features Python bindings. The extent to which the package is being maintained is somewhat unclear, but there were updates as recently as 2013 and the documentation includes installation instructions for modern operating systems such as Ubuntu. There is insufficient information available on how calibration is performed in GIPSY.

NEWSTAR

NEWSTAR[3] (Netherlands East West Synthesis Telescope Array Reduction) is a package tailored to data reduction for the WSRT (Westerbork Synthesis Radio Telescope, Noordam (1994)). NEWSTAR was designed and written by W. N. Brouw.

NEWSTAR, which was implemented primarily in Fortran, is now deprecated and is being neither developed nor maintained. This is as a result of other, more flexible reduction packages superseding it. It was, however, an important stepping stone on the path to the more modern packages. From a calibration perspective, it implemented a traditional constrained least-squares solver and was the first package to implement redundant calibration[4] (Noordam 1982; Noordam and de Bruyn 1982).

GILDAS

The first development of GILDAS[5], a software suite specialised in processing both single dish and interferometric (sub-)millimeter radio observations, began in 1983 at the Observatoire de Grenoble. It is still undergoing regular development and maintenance and is used in reducing data from the IRAM (Institut de Radioastronomie Millimétrique) 30m telescope (single dish) and the NOEMA (Northern Extended Millimeter Array) project.

The majority of GILDAS is written in Fortran although some of its components are coded in C/C++. Whilst its documentation is extensive, there does not appear to be a description of its calibration implementation. However, there are hints that it of the form described by Thompson and D'Addario (1982). The package does seems to have extensive 1GC capabilities.

Difmap

Difmap[6] implements a technique called difference mapping (hence the name), which is a highly interactive method of iteratively building up a sky model for use in the self-calibration loop (Shepherd 1997; Taylor 1994). In addition to offering user-friendly data visualisation and editing, Difmap is self-contained, allowing users to perform the entire self-calibration procedure (model construction, calibration, imaging) with a single program.

Whilst it was originally intended for use with VLBI continuum data, Difmap was also used to process VLA and other interferometer data. Difmap was written in a combination of

[3] https://github.com/lofar-astron/Newstar

[4] It also supported large-N mosaicing, full-polarisation, multi-frequency synthesis, detailed source modeling and a primordial version of the RIME.

[5] https://www.iram.fr/IRAMFR/GILDAS/

[6] ftp://ftp.eso.org/scisoft/scisoft4/sources/difmap/difmap.html

Fortran and C. The package has undergone no significant development since 1995 and should likely be considered deprecated.

Although it is not clear how the calibration component of Difmap was implemented, it most likely used a conventional least-squares approach. This assumption is based on the age of package and the fact that the implementation was based on code supplied by T. J. Cornwell (of Cornwell and Wilkinson (1981)).

Miriad

Initial development of Miriad[7] (multi-channel image reconstruction, image analysis and display) began in 1988 (Sault et al. 1995) with the aim of becoming a complete data reduction package for BIMA (Berkeley-Illinois-Maryland Association) and its associated instruments. In 1990, following a move by two of the Miriad team members to the ATNF (Australia Telescope National Facility), it was extended to address the needs of ATCA (Australia Telescope Compact Array). It is still in use, particularly in the HI (neutral hydrogen) science community and is particularly popular with ATCA users. It also seems to be undergoing regular development and maintenance.

Miriad is written in a combination of C and Fortran, much like many of the tools of this era. It has an excellent users guide (Sault and Killeen 2004) which describes its approach to both 1GC and self-calibration quite clearly although the numerical techniques used are not mentioned. Again, it is likely of the form suggested by Thompson and D'Addario (1982).

AIPS++/CASA

As previously mentioned, AIPS has slowly been replaced (although not entirely) by CASA[8] (Common Astronomy Software Applications, McMullin et al. (2007)), the package previously known as AIPS++. Its main purpose is to provide data post-processing (including calibration) for modern radio instruments.

CASA is coded in C++ with a Python (specifically iPython) front-end. Consequently it boasts the performance of the former and the flexible scripting of the latter. This flexibility is one of several reasons for its popularity. Additional reasons include its relative ease of use (compared to its predecessors) and the large number of experienced users who can offer support. Additionally, as one of the most widely used pieces of software in radio astronomy, it undergoes regular updates and maintenance.

CASA was one of the first packages to make use of an explicit form of the RIME, as noted by Smirnov (2011b), and can solve for a number of different Jones terms (a full description of which appears in Ott and Kern (2017)). Whilst the numerical method by which this is accomplished is not mentioned, it is likely a full-polarisation least squares solution, similar to that described by Hamaker (2000).

[7] http://www.atnf.csiro.au/computing/software/miriad/
[8] https://casa.nrao.edu/

MeqTrees

MeqTrees[9] is a software package specifically tailored to the rapid implementation of explicit measurement equations and their subsequent use in both simulation and calibration (Noordam and Smirnov 2010). The authors' original goal was to provide a rapid prototyping environment that would still be as performant as hand-written, optimised code. To this end, the back-end of MeqTrees is written in C++ whilst the front-end is written in Python. Measurement equations (or numerical models) are defined via a Tree Definition Language (TDL), which is itself based on Python. The package also has extensive plotting and visualisation tools. All of these factors combined allow for very rapid development/experimentation.

The package is still widely used, particularly for performing simulations of instruments such as the SKA and LOFAR. It was also the first package implemented with 3GC in mind although there are now more powerful alternatives for calibration. Internally, MeqTrees performs self-calibration using StEFCal and direction-dependent calibration (differential gain calibration) is performed using a full non-linear least squares solver (simultaneous updates to all directions).

ASKAPsoft

ASKAPsoft[10] (Australian Square Kilometre Array Pathfinder) is a collection of utilities which together form the ASKAP data processing pipeline. A detailed description of the pipeline steps appears in Cornwell et al. (2011), but those steps include extensive calibration which is well documented and explained. In this instance, the least-squares fit between the model and the data is performed by solving the normal equations (and invoking some of the original trickery of Cornwell and Wilkinson (1981) and Thompson and D'Addario (1982)). This was discussed extensively in Subsection 3.2.3.

As ASKAP is itself a new instrument (and an SKA pathfinder), its software is modern. The majority of the ASKAPsoft code base is implemented in Python and C++, both of which are quite common in modern applications in astronomy. Naturally, given how new the instrument is, the software is still undergoing active development and maintenance.

DPPP, Prefactor, and Factor

The DPPP (Default Preprocessing Pipeline) is the LOFAR pipeline processing utility which can be used to perform a number of preprocessing steps on LOFAR data. This includes calibration, specifically 1GC and 2GC, but also flagging, averaging and other common reduction steps. It is primarily implemented in C++, with a strong emphasis on speed due to the requirements of the instrument. It does, however, support the addition of user-defined steps coded in either C++ or Python.

Prefactor[11] makes use of the DPPP and its own Python scripts to implement a generic pipeline which performs all the steps necessary to make use of Factor[12], a facet-based,

[9] http://meqtrees.net/
[10] https://bitbucket.csiro.au/projects/CASSSOFT/repos/askapsoft
[11] https://github.com/lofar-astron/prefactor
[12] https://github.com/lofar-astron/factor

direction-dependent calibration scheme described by van Weeren et al. (2016). This is an extension of peeling (Noordam 2004) wherein individual gain solutions are determined per facet although the numerical method used to determine the gain solutions is not described. All of these packages are undergoing regular development and maintenance.

KillMS

The KillMS[13] package implements direction-dependent calibration using the ideas of complex optimisation. It features two different algorithms, the first of which, *CohJones* (Smirnov and Tasse 2015), is a complex non-linear least squares solver which makes assumptions about the separability of the gains. This is a fast solver, in the vein of StEFCal.

The second algorithm is called *KAFCA* (Tasse 2014b), which makes use of a Kalman filter (Kalman 1960) to tackle the direction-dependent calibration problem and estimate the physical terms of the RIME.

KillMS is written primarily in Python, with some of the most computationally intensive steps in C. The package is still undergoing development, but has already been applied to both LOFAR and MeerKAT data. In fact, it has already been adopted for the LOFAR surveys pipeline (Shimwell et al. 2019).

SAGECal

This package shares its name with one of the (several) algorithms it implements. SAGECal[14] is a collection of highly optimised calibration routines, including implementations of robust calibration, SAGECal and consensus optimisation (see Subsection 3.3.3 for the details of the algorithms). A number of numerical solvers are supported, including the Levenberg-Marquardt, LBFGS (the limited memory Broyden–Fletcher–Goldfarb–Shanno), Riemannian Trust Region and Nesterov's accelerated gradient descent algorithms.

SAGECal is implemented in a mixture of C, C++ and CUDA, and it is unique in having the greatest focus on distributed computation. This distribution is accomplished using MPI (Message Passing Interface). The package has already been demonstrated to work on LOFAR data (see the aforementioned Subsection) and seems to be well-maintained.

[13] https://github.com/saopicc/killMS
[14] https://github.com/nlesc-dirac/sagecal

Chapter 4

Mathematical Background and Context

The preceding chapters have introduced the RIME and described both the origins and current state-of-the-art in calibration. It is now necessary to describe the mathematics upon which the remainder of this work will be based.

We will begin with an introduction to the most common non-linear least squares (NLLS) solvers and their derivation. As an illustrative example, these solvers will be applied to the calibration problem using the traditional method of dealing with complex variables: treating the real and imaginary parts as independent. This will lead us to a discussion of complex optimisation and the introduction of the Wirtinger derivatives. Finally, we will establish some of the notation which will be used in subsequent chapters.

4.1 A note on notation

Before we present any more mathematics, we should first pause to establish some of the conventions. Scalar values will be represented by upper- or lower-case, italicised symbols e.g. λ, S. Vectors will be represented by lower-case, bold, non-italic symbols e.g. $\mathbf{x}$, $\mathbf{r}$. Matrices will be represented by upper-case, bold, non-italic symbols e.g. $\mathbf{J}$, $\mathbf{H}$. Conjugation, transposition and the combination of the two will be represented by $\bar{(\cdot)}$, $(\cdot)^T$ and $(\cdot)^H$ respectively.

4.2 Non-linear least squares methods

Antenna-based gain calibration in radio interferometry is fundamentally a non-linear least squares problem. As such, there are a plethora of numerical methods which can be employed to solve it. However, the most popular of these methods, the Levenberg-Marquardt (LM) algorithm (Levenberg 1944; Marquardt 1963), is sufficient for our purposes. The LM algorithm is an extension of the simpler Gauss-Newton (GN) algorithm (Madsen et al. 2004), for which we will provide a derivation.

4.2.1 Deriving the Gauss-Newton and Levenberg-Marquardt algorithms

In the interest of completeness, we will derive the GN update rule and then show how it can be extended to LM. First, let us consider the function we want to minimise:

$$f(\mathbf{r}(\mathbf{x})) = \|\mathbf{r}(\mathbf{x})\|_2^2 = \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x}), \tag{4.1}$$

where:

$$\mathbf{r}(\mathbf{x}) = \mathbf{d} - \mathbf{m}(\mathbf{x}). \tag{4.2}$$

In the above expressions, $\mathbf{r}$ is a length m vector of residual values obtained after subtracting the model vector $\mathbf{m}$ (which is a function of the length n vector of unknowns $\mathbf{x}$) from the data vector $\mathbf{d}$. The goal is to find the values of $\mathbf{x}$ which minimise f. This is accomplished by finding an update vector, $\boldsymbol{\Delta}\mathbf{x}$, which iteratively moves us closer to the solution. Consequently, Equation 4.2 can be rewritten as:

$$\mathbf{r}(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}) = \mathbf{d} - \mathbf{m}(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}). \tag{4.3}$$

The vector $\mathbf{m}(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x})$ appearing in Equation 4.3 can be approximated by its Taylor expansion:

$$\mathbf{m}(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}) \approx \mathbf{m}(\mathbf{x}) + \mathbf{J}\boldsymbol{\Delta}\mathbf{x} + \mathcal{O}(\|\boldsymbol{\Delta}\mathbf{x}\|^2), \tag{4.4}$$

where $\mathbf{J}$ is the Jacobian, an $m \times n$ matrix of partial derivatives, the entries of which are given by:

$$[\mathbf{J}]_{ab} = \frac{\partial \mathbf{m}_a}{\partial \mathbf{x}_b}. \tag{4.5}$$

Ignoring the higher order terms of Equation 4.4, we can return to Equation 4.1 and substitute the approximation, yielding:

$$\begin{aligned} f(\mathbf{r}(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x})) = [\mathbf{r}(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x})]^T[\mathbf{r}(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x})] &\approx [\mathbf{d} - \mathbf{m}(\mathbf{x}) - \mathbf{J}\boldsymbol{\Delta}\mathbf{x}]^T[\mathbf{d} - \mathbf{m}(\mathbf{x}) - \mathbf{J}\boldsymbol{\Delta}\mathbf{x}] \\ &\approx [\mathbf{r}(\mathbf{x}) - \mathbf{J}\boldsymbol{\Delta}\mathbf{x}]^T[\mathbf{r}(\mathbf{x}) - \mathbf{J}\boldsymbol{\Delta}\mathbf{x}]. \end{aligned} \tag{4.6}$$

What follows is a simplification of the right hand side of Equation 4.6:

$$\begin{aligned} [\mathbf{r}(\mathbf{x}) - \mathbf{J}\boldsymbol{\Delta}\mathbf{x}]^T[\mathbf{r}(\mathbf{x}) - \mathbf{J}\boldsymbol{\Delta}\mathbf{x}] &= \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x}) - \mathbf{r}(\mathbf{x})^T\mathbf{J}\boldsymbol{\Delta}\mathbf{x} - [\boldsymbol{\Delta}\mathbf{x}]^T\mathbf{J}^T\mathbf{r}(\mathbf{x}) + [\boldsymbol{\Delta}\mathbf{x}]^T\mathbf{J}^T\mathbf{J}\boldsymbol{\Delta}\mathbf{x} \\ &= \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x}) - 2\mathbf{r}(\mathbf{x})^T\mathbf{J}\boldsymbol{\Delta}\mathbf{x} + [\boldsymbol{\Delta}\mathbf{x}]^T\mathbf{J}^T\mathbf{J}\boldsymbol{\Delta}\mathbf{x}. \end{aligned} \tag{4.7}$$

Taking the derivative of Equation 4.7 with respect to $\boldsymbol{\Delta}\mathbf{x}$ and setting it to zero (standard procedure for finding the stationary points of a function) we obtain:

$$-2\mathbf{J}^T\mathbf{r}(\mathbf{x}) + 2\mathbf{J}^T\mathbf{J}\boldsymbol{\Delta}\mathbf{x} = 0, \tag{4.8}$$

which, with some rearrangement and simplification, yields the Gauss-Newton update rule:

$$\boldsymbol{\Delta}\mathbf{x} = [\mathbf{J}^T\mathbf{J}]^{-1}\mathbf{J}^T\mathbf{r}(\mathbf{x}). \tag{4.9}$$

It is interesting to note that this update is relatively simple - it depends only on the Jacobian matrix and the residual vector. The inverse term, the product $\mathbf{J}^T\mathbf{J}$, is an approximation of the Hessian (matrix of second-order partial derivatives) which contains curvature information. There are also similarities between this expression and result of applying the normal equations (see Subsection 3.2.3).

Having derived the Gauss-Newton update, it is possible to introduce the modifications presented by Levenberg (Levenberg 1944) and Marquardt (Marquardt 1963). Levenberg suggested a damped version of the same update rule:

$$\mathbf{\Delta x} = [\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}]^{-1}\mathbf{J}^T\mathbf{r}(\mathbf{x}), \tag{4.10}$$

where $\mathbf{I}$ is an $n \times n$ identity matrix and λ is a scalar damping parameter. This λ value is usually determined heuristically, but it does have an intuitive interpretation. As $\lambda \to 0$, the update becomes equivalent to Gauss-Newton whereas for $\lambda \to \infty$ it is approximately a method of steepest descent (Madsen et al. 2004). Thus, the λ value controls the degree of hybridisation between the two methods. Generally, methods of steepest descent are slow to converge near the solution but perform well during the initial iterations. As such, λ will usually decrease as the algorithm approaches the solution.

Equation 4.10 is not without its flaws. As λ becomes large $[\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}]^{-1} \approx \lambda^{-1}\mathbf{I}$ which does nothing other than scale the result of $\mathbf{J}^T\mathbf{r}(\mathbf{x})$. Marquardt's contribution was to replace the identity matrix with a matrix containing the diagonal entries of $\mathbf{J}^T\mathbf{J}$:

$$\mathbf{\Delta x} = [\mathbf{J}^T\mathbf{J} + \lambda\mathrm{diag}(\mathbf{J}^T\mathbf{J})]^{-1}\mathbf{J}^T\mathbf{r}(\mathbf{x}). \tag{4.11}$$

This alteration incorporates the curvature information present in $\mathbf{J}^T\mathbf{J}$ by providing appropriate scaling i.e. step sizes will be larger in directions of low gradient (low curvature) which aids convergence close to the solution. Equation 4.11 is the Levenberg-Marquardt update rule.

One final note before moving on to an example: these algorithms only guarantee local convergence and as such, depending on the behaviour of the problem in question, some care must be taken with the solutions. Fortunately, it is usually possible to infer the relative success of the algorithms by performing some diagnostic checks on the results.

4.2.2 Calibration as a least squares problem

Having introduced the non-linear least squares methods which we will be using, it is interesting to apply them to the calibration problem using the conventional approach of splitting the problem into its real and imaginary parts. For the sake of simplicity, this example will be limited to the unweighted, scalar (single correlation) case. The objective function S is given by:

$$S = \sum_{p \neq q} |r_{pq}|^2 = \sum_{p \neq q} |d_{pq} - v_{pq}|^2 = \sum_{p \neq q} |d_{pq} - g_p m_{pq} \bar{g}_q|^2, \tag{4.12}$$

where r_{pq} is the residual on baseline pq obtained by subtracting the product (v_{pq}) of the predicted visibility m_{pq} and its associated gains (g_p and $\bar{g}_q$) from the observed visibility d_{pq}.

In order to proceed, we need expressions for the components of the Gauss-Newton and Levenberg-Marquardt algorithms, specifically the residual vector and the Jacobian. The residual vector is straightforward to define and we obtain:

$$\mathbf{r} = \left[r_{pq} \right] \Big\} \; {\scriptstyle [pq]=1,\ldots,N_{\text{BL}} \text{ s.t. } (p<q)}. \tag{4.13}$$

The compound index pq enumerates the N_{BL} baselines for which $p < q$ e.g. for the three antenna, three baseline case $pq \in [12, 13, 23]$. The model vector ($\mathbf{v}$) shares both its shape and indexing with $\mathbf{r}$.

In order to compute the Jacobian, one would usually differentiate the residual vector with respect to a vector of unknowns. In this case, that vector contains the N_{A} complex-valued antenna gains given by:

$$\mathbf{g} = \left[g_p \right] \Big\} \; {\scriptstyle [p]=1,\ldots,N_{\text{A}}}. \tag{4.14}$$

It just so happens that in the field of radio interferometry the Jacobian is often defined as the derivative of the model vector ($\mathbf{v}$) with respect to the gains ($\mathbf{g}$) instead of the residual vector ($\mathbf{r}$) with respect to the gains. Whilst this does introduce a change of sign, the two approaches are otherwise completely equivalent and we will stick with the convention by defining the entries of the Jacobian as:

$$[\mathbf{J}]_{ab} = \left[\frac{\partial v_a}{\partial g_b} \right]. \tag{4.15}$$

However, it is at this point that we run into the problem which necessitates splitting the problem into its real and imaginary parts - the derivative $\frac{\partial \bar{g}_p}{\partial g_p}$ does not exist in the context of conventional calculus.

This problem is usually circumvented by defining the following augmented vector equivalents for $\mathbf{r}$, $\mathbf{v}$ and $\mathbf{g}$:

$$\breve{\mathbf{r}} = \begin{bmatrix} \mathbf{r}^R \\ \mathbf{r}^I \end{bmatrix} = \begin{bmatrix} r^R_{pq} \\ r^I_{pq} \end{bmatrix} \begin{matrix} \Big\} \; {\scriptstyle [pq]=1,\ldots,N_{\text{BL}} \text{ s.t. } (p<q)} \\ \Big\} \; {\scriptstyle [pq]=1,\ldots,N_{\text{BL}} \text{ s.t. } (p<q)} \end{matrix}, \tag{4.16}$$

$$\breve{\mathbf{v}} = \begin{bmatrix} \mathbf{v}^R \\ \mathbf{v}^I \end{bmatrix} = \begin{bmatrix} v^R_{pq} \\ v^I_{pq} \end{bmatrix} \begin{matrix} \Big\} \; {\scriptstyle [pq]=1,\ldots,N_{\text{BL}} \text{ s.t. } (p<q)} \\ \Big\} \; {\scriptstyle [pq]=1,\ldots,N_{\text{BL}} \text{ s.t. } (p<q)} \end{matrix}, \tag{4.17}$$

$$\breve{\mathbf{g}} = \begin{bmatrix} \mathbf{g}^R \\ \mathbf{g}^I \end{bmatrix} = \begin{bmatrix} g^R_{p} \\ g^I_{p} \end{bmatrix} \begin{matrix} \Big\} \; {\scriptstyle [p]=1,\ldots,N_{\text{A}}} \\ \Big\} \; {\scriptstyle [p]=1,\ldots,N_{\text{A}}} \end{matrix}, \tag{4.18}$$

where $(\cdot)^R$ denotes the real part and $(\cdot)^I$ denotes the imaginary part of their associated quantities. Note that these vectors are all twice the length of their unaugmented equivalents and the problem of finding the N_{A} complex-valued antenna gains has been replaced with finding $2N_{\text{A}}$ real values.

The Jacobian's dimensions are also doubled by the change and it becomes possible to identify four unique blocks within the Jacobian:

$$\mathbf{J} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \quad \text{where} \quad \begin{cases} [\mathbf{A}]_{ab} = \left[\dfrac{\partial v_a^R}{\partial g_b^R}\right] \\ [\mathbf{B}]_{ab} = \left[\dfrac{\partial v_a^R}{\partial g_b^I}\right] \\ [\mathbf{C}]_{ab} = \left[\dfrac{\partial v_a^I}{\partial g_b^R}\right] \\ [\mathbf{D}]_{ab} = \left[\dfrac{\partial v_a^I}{\partial g_b^I}\right] \end{cases}. \tag{4.19}$$

Each block is an $N_{\mathrm{BL}} \times N_{\mathrm{A}}$ matrix, with entries at row a, column b (within a block) given above. Recall that each entry of $\mathbf{v}$ is associated with a specific baseline, i.e. $a \to pq$.

This result is not yet particularly useful from an analytic perspective, mainly because each v_{pq} is a product of several complex numbers. Thus, splitting it into its real and imaginary parts requires some additional work. Returning to the definition of v_{pq}, we have:

$$\begin{aligned} v_{pq} = g_p m_{pq} \bar{g}_q &= (g_p^R + i g_p^I)(m_{pq}^R + i m_{pq}^I)(g_q^R - i g_q^I) \\ &= (g_p^R m_{pq}^R g_q^R + g_p^R m_{pq}^I g_q^I + g_p^I m_{pq}^R g_q^I - g_p^I m_{pq}^I g_q^R) + \\ &\quad i(g_p^I m_{pq}^R g_q^R - g_p^R m_{pq}^R g_q^I + g_p^R m_{pq}^I g_q^R + g_p^I m_{pq}^I g_q^I), \end{aligned} \tag{4.20}$$

and consequently:

$$v_{pq}^R = g_p^R m_{pq}^R g_q^R + g_p^R m_{pq}^I g_q^I + g_p^I m_{pq}^R g_q^I - g_p^I m_{pq}^I g_q^R, \tag{4.21}$$

$$v_{pq}^I = g_p^I m_{pq}^R g_q^R - g_p^R m_{pq}^R g_q^I + g_p^R m_{pq}^I g_q^R + g_p^I m_{pq}^I g_q^I. \tag{4.22}$$

Using these expressions, it is suddenly possible to write down the per-block analytic derivatives as:

$$[\mathbf{A}]_{ab} = \frac{\partial v_{a\to pq}^R}{\partial g_b^R} = (m_{pq}^R g_q^R + m_{pq}^I g_q^I)\delta_p^b + (g_p^R m_{pq}^R - g_p^I m_{pq}^I)\delta_q^b, \tag{4.23}$$

$$[\mathbf{B}]_{ab} = \frac{\partial v_{a\to pq}^R}{\partial g_b^I} = (m_{pq}^R g_q^I - m_{pq}^I g_q^R)\delta_p^b + (g_p^R m_{pq}^I + g_p^I m_{pq}^R)\delta_q^b, \tag{4.24}$$

$$[\mathbf{C}]_{ab} = \frac{\partial v_{a\to pq}^I}{\partial g_b^R} = (-m_{pq}^R g_q^I + m_{pq}^I g_q^R)\delta_p^b + (g_p^R m_{pq}^I + g_p^I m_{pq}^R)\delta_q^b, \tag{4.25}$$

$$[\mathbf{D}]_{ab} = \frac{\partial v_{a\to pq}^I}{\partial g_b^I} = (m_{pq}^R g_q^R + m_{pq}^I g_q^I)\delta_p^b + (-g_p^R m_{pq}^R + g_p^I m_{pq}^I)\delta_q^b, \tag{4.26}$$

where the Dirac delta has its usual meaning ($\delta = 1$ if $p = b$ else $\delta = 0$). A close inspection of these expressions reveals that the terms in block $\mathbf{A}$ are very similar to those in block $\mathbf{D}$. The same is true of blocks $\mathbf{B}$ and $\mathbf{C}$.

From a numerical perspective, this definition of the Jacobian is all that is necessary to

compute a parameter update - it can simply be substituted, along with the residual vector, into the relevant update rule. However, we will take this analytic derivation a little further in order to show some interesting links between it and other approaches. To do so, an analytic expression for transpose of the Jacobian is also necessary. It is given by:

$$\mathbf{J}^T = \begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \\ \mathbf{B}^T & \mathbf{D}^T \end{bmatrix}, \tag{4.27}$$

where the contents of the blocks are:

$$[\mathbf{A}^T]_{ab} = (m^R_{pq} g^R_q + m^I_{pq} g^I_q)\delta^a_p + (g^R_p m^R_{pq} - g^I_p m^I_{pq})\delta^a_q, \tag{4.28}$$
$$[\mathbf{B}^T]_{ab} = (m^R_{pq} g^I_q - m^I_{pq} g^R_q)\delta^a_p + (g^R_p m^I_{pq} + g^I_p m^R_{pq})\delta^a_q, \tag{4.29}$$
$$[\mathbf{C}^T]_{ab} = (-m^R_{pq} g^I_q + m^I_{pq} g^R_q)\delta^a_p + (g^R_p m^I_{pq} + g^I_p m^R_{pq})\delta^a_q, \tag{4.30}$$
$$[\mathbf{D}^T]_{ab} = (m^R_{pq} g^R_q + m^I_{pq} g^I_q)\delta^a_p + (-g^R_p m^R_{pq} + g^I_p m^I_{pq})\delta^a_q. \tag{4.31}$$

The important difference is that the transposition swaps the interpretation of the axes. Now each column (b) in a block is associated with a specific baseline ($b \rightarrow pq$) and the derivatives are taken along the row axis (a).

We are now in a position to begin substituting and combining our analytic expressions. Taking the product of the transpose of the Jacobian and the residual vector gives:

$$\mathbf{J}^T \breve{\mathbf{r}} = \begin{bmatrix} \mathbf{A}^T \mathbf{r}^R + \mathbf{C}^T \mathbf{r}^I \\ \mathbf{B}^T \mathbf{r}^R + \mathbf{D}^T \mathbf{r}^I \end{bmatrix} = \begin{bmatrix} \mathbf{E} \\ \mathbf{F} \end{bmatrix}, \tag{4.32}$$

where:

$$[\mathbf{E}]_a = \sum_{q \neq a} (m^R_{aq} g^R_q + m^I_{aq} g^I_q) r^R_{aq} + (m^I_{aq} g^R_q - m^R_{aq} g^I_q) r^I_{aq}, \tag{4.33}$$
$$[\mathbf{F}]_a = \sum_{q \neq a} (m^R_{aq} g^I_q - m^I_{aq} g^R_q) r^R_{aq} + (m^R_{aq} g^R_q + m^I_{aq} g^I_q) r^I_{aq}. \tag{4.34}$$

This step may seem confusing, but it can be verified by expanding the above expressions and noting that $m^I_{pq} = -m^I_{qp}$. The resulting vector has $2N_\mathrm{A}$ entries, with the top and bottom halves being assosciated with the updates to the real and imaginary components respectively.

Writing out similar analytic expressions for $\mathbf{J}^T\mathbf{J}$ is possible but rapidly becomes messy. Instead, we will examine its basic structure. Like the Jacobian, it can be treated as having four distinct blocks:

$$\mathbf{J}^T\mathbf{J} = \begin{bmatrix} \mathbf{A}' & \mathbf{C}' \\ \mathbf{B}' & \mathbf{D}' \end{bmatrix}, \tag{4.35}$$

where each block is substantially more complicated than those of the Jacobian. We can, however, make a simplifying assumption - that it can be approximated by a diagonal matrix. This assumption essentially neglects covariance between the real and imaginary components. Whilst this may increase the number of iterations required for the methods to converge, it

also makes the inversion of $\mathbf{J}^T\mathbf{J}$ much cheaper. We are left with:

$$\mathbf{J}^T\mathbf{J} = \begin{bmatrix} \mathrm{diag}(\mathbf{A}') & \mathbf{0} \\ \mathbf{0} & \mathrm{diag}(\mathbf{D}') \end{bmatrix}, \tag{4.36}$$

where:

$$\mathrm{diag}(\mathbf{A}') = [\mathbf{A}']_{aa} = \sum_{q\neq a}(m^R_{aq}g^R_q + m^I_{aq}g^I_q)^2 + (m^I_{aq}g^R_q - m^R_{aq}g^I_q)^2, \tag{4.37}$$

$$\mathrm{diag}(\mathbf{D}') = [\mathbf{D}']_{aa} = \sum_{q\neq a}(m^R_{aq}g^I_q - m^I_{aq}g^R_q)^2 + (m^R_{aq}g^R_q + m^I_{aq}g^I_q)^2. \tag{4.38}$$

These expressions are in fact equivalent. Combining the analytic expressions for $\mathbf{J}^T\mathbf{J}$ with those for $\mathbf{J}^T\breve{\mathbf{r}}$ we can write down the per-antenna, per-component, Gauss-Newton update rule:

$$\boldsymbol{\Delta}\mathbf{g}^R_a = \begin{bmatrix} \sum_{q\neq a}(m^R_{aq}g^R_q + m^I_{aq}g^I_q)^2 + (m^I_{aq}g^R_q - m^R_{aq}g^I_q)^2 \end{bmatrix}^{-1}, \\ \sum_{q\neq a}(m^R_{aq}g^R_q + m^I_{aq}g^I_q)r^R_{aq} + (m^I_{aq}g^R_q - m^R_{aq}g^I_q)r^I_{aq} \tag{4.39}$$

$$\boldsymbol{\Delta}\mathbf{g}^I_a = \begin{bmatrix} \sum_{q\neq a}(m^R_{aq}g^R_q + m^I_{aq}g^I_q)^2 + (m^I_{aq}g^R_q - m^R_{aq}g^I_q)^2 \end{bmatrix}^{-1}. \\ \sum_{q\neq a}(m^R_{aq}g^I_q - m^I_{aq}g^R_q)r^R_{aq} + (m^R_{aq}g^R_q + m^I_{aq}g^I_q)r^I_{aq} \tag{4.40}$$

Whilst these expressions are perfectly adequate, they are somewhat clumsy and still rely on splitting the problem. We can do better by returning to the ideas of Thompson and D'Addario (1982) and writing:

$$\boldsymbol{\Delta}\mathbf{g}_a = \boldsymbol{\Delta}\mathbf{g}^R_a + i\boldsymbol{\Delta}\mathbf{g}^I_a. \tag{4.41}$$

It is immediately possible to pull out the common inverse term, and note that:

$$\begin{bmatrix} \sum_{q\neq a}(m^R_{aq}g^R_q + m^I_{aq}g^I_q)^2 + (m^I_{aq}g^R_q - m^R_{aq}g^I_q)^2 \end{bmatrix}^{-1} = \begin{bmatrix} \sum_{q\neq a} m_{aq}\bar{g}_q g_q \bar{m}_{aq} \end{bmatrix}^{-1}. \tag{4.42}$$

This can be verified by re-splitting the right hand side of the equation. What remains is then the $\mathbf{J}^T\breve{\mathbf{r}}$ component:

$$\sum_{q\neq a}(m^R_{aq}g^R_q + m^I_{aq}g^I_q)r^R_{aq} + (m^I_{aq}g^R_q - m^R_{aq}g^I_q)r^I_{aq} \\ + i\sum_{q\neq a}(m^R_{aq}g^I_q - m^I_{aq}g^R_q)r^R_{aq} + (m^R_{aq}g^R_q + m^I_{aq}g^I_q)r^I_{aq} = \sum_{q\neq a} r_{aq}g_q\bar{m}_{aq}. \tag{4.43}$$

The deluge of simplification required to show the above has been omitted for the sake of brevity. It can be easily verified by re-splitting the result. Combining all the simplification

we have:

$$\mathbf{\Delta g}_a = \left[\sum_{q\neq a} m_{aq}\bar{g}_q g_q \bar{m}_{aq}\right]^{-1} \sum_{q\neq a} r_{aq} g_q \bar{m}_{aq}. \tag{4.44}$$

Finally, if we substitute $r_{aq} = d_{aq} - g_a m_{aq}\bar{g}_q$ we can show, following an observation made in both Tasse (2014a) and Smirnov and Tasse (2015), that Equation 4.44 reduces to:

$$\mathbf{g}_a = \left[\sum_{q\neq a} m_{aq}\bar{g}_q g_q \bar{m}_{aq}\right]^{-1} \sum_{q\neq a} d_{aq} g_q \bar{m}_{aq}. \tag{4.45}$$

This is very simply the scalar version of StEFCal (Salvini and Wijnholds 2014), which of course means it is also very similar to the method proposed in Hamaker (2000). This means that it could have been derived using the normal equation method. The crucial take-away from this section should be how closely related some of the most fundamental methods of calibration are.

4.3 Complex Optimisation

Having demonstrated how non-linear least squares has traditionally been applied to the radio interferometric calibration problem, the stage is now set for the introduction of complex optimisation and the Wirtinger derivatives. Much of what follows will be drawn from Kenyon et al. (2018), the paper associated with this work, and Smirnov and Tasse (2015), its antecedent.

4.3.1 Non-linear least squares for scalar complex variables

It has already been mentioned that the derivative $\frac{\partial \bar{g}_p}{\partial g_p}$, and in fact the derivative of any quantity with respect to its conjugate, does not exist using the conventional definitions of calculus. Fortunately, the Wirtinger derivatives (Wirtinger 1927):

$$\frac{\partial}{\partial z} = \frac{1}{2}\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right), \quad \frac{\partial}{\partial \bar{z}} = \frac{1}{2}\left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right), \tag{4.46}$$

provide a means for circumventing this problem by treating the complex-valued variable ($z = x + iy$) and its conjugate ($\bar{z} = x - iy$) as independent. Consequently, it is simple to show that:

$$\frac{\partial z}{\partial \bar{z}} = \frac{\partial \bar{z}}{\partial z} = 0. \tag{4.47}$$

Kreutz-Delgado (2009) and Sorber et al. (2012) make use of the Wirtinger derivatives to extend conventional non-linear least squares solvers to function with complex variables, including the Gauss-Newton and Levenberg-Marquardt algorithms. We will not present their derivations and instead merely present the resulting complex equivalents of the familiar update rules:

$$\mathbf{\Delta \check{z}} = [\mathbf{J}^H\mathbf{J}]^{-1}\mathbf{J}^H\check{\mathbf{r}}(\check{\mathbf{z}}), \tag{4.48}$$

$$\mathbf{\Delta \check{z}} = [\mathbf{J}^H\mathbf{J} + \lambda \mathrm{diag}(\mathbf{J}^H\mathbf{J})]^{-1}\mathbf{J}^H\check{\mathbf{r}}(\check{\mathbf{z}}), \tag{4.49}$$

where all symbols retain their previous definitions. The two important distinctions between these expressions and their more conventional equivalents is the conjugate transpose operator, $(\cdot)^H$, replacing transposition and the use of the augmented vectors denoted by $\breve{(\cdot)}$. However, unlike the augmented vectors described in Subsection 4.2.2, these are formed by augmenting the quantity in question with its conjugate.

Thus, for a length n vector of complex variables, $\mathbf{z}$, we define our length $2n$ augmented parameter vector as:

$$\breve{\mathbf{z}} = \begin{bmatrix} \mathbf{z} \\ \bar{\mathbf{z}} \end{bmatrix}. \tag{4.50}$$

Similarly, for a length m residual vector, $\mathbf{r}(\breve{\mathbf{z}})$ (noting that it may depend on the entire augmented parameter vector), we define the length $2m$ augmented residual vector as:

$$\breve{\mathbf{r}} = \begin{bmatrix} \mathbf{r}(\breve{\mathbf{z}}) \\ \bar{\mathbf{r}}(\breve{\mathbf{z}}) \end{bmatrix}. \tag{4.51}$$

As was mentioned in the derivation of the real-imaginary case, radio-interferometric convention often results in the differentiation of the model vector. Consequently, for a length m model vector $\mathbf{v}(\breve{\mathbf{z}})$, we define the length $2m$ augmented model vector as:

$$\breve{\mathbf{v}} = \begin{bmatrix} \mathbf{v}(\breve{\mathbf{z}}) \\ \bar{\mathbf{v}}(\breve{\mathbf{z}}) \end{bmatrix}. \tag{4.52}$$

Armed with these definitions, we can write down the Jacobian as:

$$\mathbf{J} = \frac{\partial \breve{\mathbf{v}}(\breve{\mathbf{z}})}{\partial \breve{\mathbf{z}}} = \begin{bmatrix} \dfrac{\partial \mathbf{v}}{\partial \mathbf{z}} & \dfrac{\partial \mathbf{v}}{\partial \bar{\mathbf{z}}} \\ \dfrac{\partial \bar{\mathbf{v}}}{\partial \mathbf{z}} & \dfrac{\partial \bar{\mathbf{v}}}{\partial \bar{\mathbf{z}}} \end{bmatrix}, \tag{4.53}$$

where we have dropped the explicit dependence of $\mathbf{v}$ on $\breve{\mathbf{z}}$ to keep the notation simple. Each entry in the Jacobian, hereafter referred to as a partial Jacobian, is an $m \times n$ block and the overall Jacobian is an $2m \times 2n$ matrix.

The quantities which have been presented thus far are all generic (i.e. are not unique to the radio interferometric calibration problem) and can be used to solve any complex non-linear least squares problem. The next chapter will deal with the application of the mathematics presented here to the specific problem of gain calibration.

4.3.2 Non-linear least squares for 2×2 complex variables

In the current era of radio interferometry, it is rare that observations contain less than four correlations. Being able to handle multiple correlations (and implicitly the polarisation information they carry) is of the utmost importance. Thus, it is sensible to establish both a method and a notation for this case.

Smirnov and Tasse (2015) developed an operator calculus for precisely this purpose. They define a matrix operator as a function, $\mathcal{F}$, which maps one complex-valued 2×2 matrix to

another:

$$\mathcal{F} : \mathbb{C}^{2\times 2} \to \mathbb{C}^{2\times 2}. \tag{4.54}$$

In other words, the application of an operator to a matrix produces a new matrix. Two such operators will be particularly useful throughout the following chapter; the left- and right-multiply operators. Given 2×2 matrices $\mathbf{A}$ and $\mathbf{B}$, the left- and right-multiply operators are given by:

$$\mathcal{L}_{\mathbf{A}}\mathbf{B} = \mathbf{A}\mathbf{B}, \tag{4.55}$$

$$\mathcal{R}_{\mathbf{A}}\mathbf{B} = \mathbf{B}\mathbf{A}. \tag{4.56}$$

We will not repeat the detailed derivation of these operators (see Appendix B of Smirnov and Tasse (2015)) but we will note some of their important properties and attempt to provide a measure of intuition regarding their use.

Firstly, let us note that there exists an isomorphism $\mathbb{W}$ which allows us to express our 2×2 operators acting on 2×2 matrices as a product between a 4×4 matrix and a vector:

$$\text{vec}(\mathcal{L}_{\mathbf{A}}\mathbf{B}) \equiv \text{vec}(\mathbf{A}\mathbf{B}) \equiv \mathbb{W}\mathcal{L}_{\mathbf{A}}\text{vec}(\mathbf{B}), \tag{4.57}$$

$$\text{vec}(\mathcal{R}_{\mathbf{A}}\mathbf{B}) \equiv \text{vec}(\mathbf{B}\mathbf{A}) \equiv \mathbb{W}\mathcal{R}_{\mathbf{A}}\text{vec}(\mathbf{B}), \tag{4.58}$$

where:

$$\mathbb{W}\mathcal{L}_{\mathbf{A}} = \mathbf{I} \otimes \mathbf{A}, \tag{4.59}$$

$$\mathbb{W}\mathcal{R}_{\mathbf{A}} = \mathbf{A}^T \otimes \mathbf{I}. \tag{4.60}$$

The origin of this isomorphism becomes quite obvious when these expressions are compared to Equation 3.19. It is sometimes necessary to exploit this isomorphism when the 2×2 operators are involved in more complicated expressions.

Naturally, there are occasions when it will be necessary to combine several operators. The following rules apply when several left- and right-multiply operators appear together:

$$\mathcal{L}_{\mathbf{A}}\mathcal{L}_{\mathbf{B}} \equiv \mathcal{L}_{\mathbf{A}\mathbf{B}}, \qquad \mathcal{R}_{\mathbf{A}}\mathcal{R}_{\mathbf{B}} \equiv \mathcal{R}_{\mathbf{B}\mathbf{A}}, \qquad \mathcal{L}_{\mathbf{A}}\mathcal{R}_{\mathbf{B}} \equiv \mathcal{R}_{\mathbf{B}}\mathcal{L}_{\mathbf{A}}. \tag{4.61}$$

Additionally, we will occasionally need to apply the inverse of an operator. We require that:

$$(\mathcal{L}_{\mathbf{A}})^{-1}\mathcal{L}_{\mathbf{A}}\mathbf{B} = \mathbf{B}, \qquad (\mathcal{R}_{\mathbf{A}})^{-1}\mathcal{R}_{\mathbf{A}}\mathbf{B} = \mathbf{B}, \tag{4.62}$$

from which it is clear that:

$$(\mathcal{L}_{\mathbf{A}})^{-1} \equiv \mathcal{L}_{\mathbf{A}^{-1}}, \qquad (\mathcal{R}_{\mathbf{A}})^{-1} \equiv \mathcal{R}_{\mathbf{A}^{-1}}. \tag{4.63}$$

The next important point is the role these operators play in the differentiation of 2×2 matrix expressions. We want to differentiate the product of 2×2 matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ with

respect to $\mathbf{A}$, $\mathbf{B}$ or $\mathbf{C}$. For now, we will concentrate on differentiating with respect to $\mathbf{A}$. This constitutes matrix-by-matrix differentiation which is not meaningful. The conventional approach to this problem is to vectorise both the product and the matrix with respect to which we are differentiating. This, in addition to the use of the somewhat ubiquitous Equation 3.19, gives:

$$\frac{\partial \text{vec}(\mathbf{ABC})}{\partial \text{vec}(\mathbf{A})} = \frac{\partial((\mathbf{BC})^T \otimes \mathbf{I})\text{vec}(\mathbf{A})}{\partial \text{vec}(\mathbf{A})} = (\mathbf{BC})^T \otimes \mathbf{I} = \mathbb{W}\mathcal{R}_{\mathbf{BC}}. \tag{4.64}$$

The result of performing the vectorised differentiation is the 4×4 matrix form of the right-multiply operator. This means we can avoid the process of vectorisation and note that in the 2×2 operator form the derivatives are given by:

$$\frac{\partial(\mathbf{ABC})}{\partial \mathbf{A}} = \mathcal{R}_{\mathbf{BC}}, \qquad \frac{\partial(\mathbf{ABC})}{\partial \mathbf{B}} = \mathcal{L}_{\mathbf{A}}\mathcal{R}_{\mathbf{C}}, \qquad \frac{\partial(\mathbf{ABC})}{\partial \mathbf{C}} = \mathcal{L}_{\mathbf{AB}}, \tag{4.65}$$

where the derivatives with respect to $\mathbf{B}$ and $\mathbf{C}$ were determined in an identical fashion to that of $\mathbf{A}$.

We are now equipped with all the tools necessary to reformulate the non-linear least squares problem of Subsection 4.3.1 in terms of 2×2 matrices. However, we will first introduce a concept which will underpin much of the following discussion; nested vectors and matrices.

Nested vectors and matrices are vectors and matrices the entries of which are themselves vectors or matrices. This will become more intuitive as we introduce specific nested terms. In general, we will notate nested quantities by adding a vector symbol $\vec{(\cdot)}$.

Our first example of a nested vector is the length n parameter vector $\vec{\mathbf{z}}$, the entries of which are potentially complex-valued 2×2 matrices:

$$\vec{\mathbf{z}} = \left[\mathbf{Z}_1, \ldots, \mathbf{Z}_n\right]^T. \tag{4.66}$$

In order to define the augmented parameter vector, we also need a way to express the element-wise conjugate transpose of a nested vector. Consequently, we define:

$$\vec{\mathbf{z}}^\dagger = \left[\mathbf{Z}_1^H, \ldots, \mathbf{Z}_n^H\right]^T. \tag{4.67}$$

We now define the 2×2 equivalents of all the augmented vectors used in Subsection 4.3.1, noting that the augmented vectors are themselves comprised of nested vectors:

$$\breve{\mathbf{z}} = \begin{bmatrix} \vec{\mathbf{z}} \\ \vec{\mathbf{z}}^\dagger \end{bmatrix}, \qquad \breve{\mathbf{r}} = \begin{bmatrix} \vec{\mathbf{r}}(\breve{\mathbf{z}}) \\ \vec{\mathbf{r}}^\dagger(\breve{\mathbf{z}}) \end{bmatrix}, \qquad \breve{\mathbf{v}} = \begin{bmatrix} \vec{\mathbf{v}}(\breve{\mathbf{z}}) \\ \vec{\mathbf{v}}^\dagger(\breve{\mathbf{z}}) \end{bmatrix}. \tag{4.68}$$

The Jacobian can now be written in terms of these augmented vectors:

$$\mathbf{J} = \frac{\partial \breve{\mathbf{v}}(\breve{\mathbf{z}})}{\partial \breve{\mathbf{z}}} = \begin{bmatrix} \dfrac{\partial \vec{\mathbf{v}}}{\partial \vec{\mathbf{z}}} & \dfrac{\partial \vec{\mathbf{v}}}{\partial \vec{\mathbf{z}}^\dagger} \\ \dfrac{\partial \vec{\mathbf{v}}^\dagger}{\partial \vec{\mathbf{z}}} & \dfrac{\partial \vec{\mathbf{v}}^\dagger}{\partial \vec{\mathbf{z}}^\dagger} \end{bmatrix}, \tag{4.69}$$

noting that each partial Jacobian is an $m \times n$ block of 2×2 entries and that the overall Jacobian therefore has shape $2m \times 2n$. Of course, the derivatives are expressed in terms of nested vectors which implies matrix-by-matrix differentiation and the results of Equation 4.64 are needed. Thus, the entries of the Jacobian will in fact be comprised of 2×2 operators.

We reiterate that these results are generic - they could be used for any 2×2 non-linear least squares problem with complex variables. This concludes the preparatory mathematics required for the derivation of the various solvers in the subsequent chapter.

Chapter 5

Solvers

The previous chapter has equipped us with all the mathematics required for deriving complex non-linear least squares solvers for radio interferometric gain calibration. In this chapter we will examine several such solvers, including a very general Jones chain solver and some specialised solvers.

The majority of this chapter will follow Kenyon et al. (2018), which is itself based upon the derivations of Smirnov and Tasse (2015).

5.1 Calibrating a Jones chain

Smirnov and Tasse (ibid.) presented the details of performing calibration with respect to a single complex gain term. Instead of rehashing their excellent derivation, we will instead extend it to the case where there are several gain terms which form a Jones chain (see Chapter 2). This will be a powerful tool for dealing with common calibration procedures such as the simultaneous solution of a bandpass ($\mathbf{B}$) and a complex gain ($\mathbf{G}$).

Whilst we will begin with the derivation for a relatively simple case, omitting factors such as direction-dependence and solution intervals, we will slowly build up to an update rule which can accommodate these requirements. Additionally, we will only consider the 2×2 Jones term case - the scalar version is merely a simplification of this more general approach.

5.1.1 Direction-independent chains

The first step in formulating the calibration problem for a generic Jones chain is to define the 2×2 RIME for baseline pq:

$$\mathbf{D}_{pq} = \left(\prod_{n=1}^{N_J} \mathbf{G}_p^{(n)} \right) \mathbf{M}_{pq} \left(\prod_{n=1}^{N_J} \mathbf{G}_q^{(n)} \right)^H + \mathbf{N}_{pq}, \tag{5.1}$$

where $\mathbf{D}_{pq}$ is the visibility observed by baseline pq, $\mathbf{M}_{pq}$ is the predicted visibility associated with baseline pq and $\mathbf{N}_{pq}$ is an additive noise term. The N_J direction-independent Jones terms (or gains) associated with antenna p are represented by the product of $\mathbf{G}_p$ terms. The same is true for the N_J gains associated with antenna q. We have chosen to use $\mathbf{G}$ to represent an arbitrary Jones term to avoid any confusion with the Jacobian.

Looking ahead a little, it is useful to clarify the interaction of the conjugate transpose with the products of several Jones terms. We have:

$$\left(\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_q^{(n)}\right)^H \equiv \left(\prod_{n=N_\mathrm{J}}^{1} \mathbf{G}_q^{(n)H}\right). \tag{5.2}$$

Given the RIME presented in Equation 5.1, the problem we wish to solve can be expressed as the following minimisation:

$$\min_{\{\mathbf{G}_p^{(n)}\}} \sum_{pq} \|\mathbf{R}_{pq}\|_F, \qquad \text{where} \qquad \mathbf{R}_{pq} = \mathbf{D}_{pq} - \mathbf{V}_{pq}, \tag{5.3}$$

and,

$$\mathbf{V}_{pq} = \left(\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_p^{(n)}\right) \mathbf{M}_{pq} \left(\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_q^{(n)}\right)^H. \tag{5.4}$$

In the above expressions, $\mathbf{R}_{pq}$ is the residual on baseline pq obtained by subtracting the model ($\mathbf{V}_{pq}$) from the observed visibilities ($\mathbf{D}_{pq}$). The components of $\mathbf{V}_{pq}$ retain their previous definitions and $\|(\cdot)\|_F$ denotes the Frobenius norm.

Although we have formulated the problem in terms of N_J Jones terms, it is important to emphasise that we will not be minimising with respect to all of the terms simultaneously. We will instead address the problem of updating a single, arbitrarily chosen term within the chain. This amounts to solving for a subset of the parameters. This may have implications for convergence; the non-linear least squares methods under consideration only assure local convergence, and solving for a single element in the chain may not yield the same minima as solving the entire problem simultaneously. Consequently, as was mentioned in Chapter 4, some care must be taken with the results. In fact, even if we were to solve for all the unknowns simultaneously, we could not guarantee convergence to a global minima.

5.1.2 Building blocks - the Jacobian

Given the importance of the Jacobian within the Gauss-Newton and Levenberg-Marquardt algorithms, it is necessary to dedicate some time to its formulation. First and foremost, let us establish the form of derivatives of $\mathbf{V}_{pq}$ with respect to the j'th Jones term and its conjugate as:

$$\frac{\partial \mathbf{V}_{pq}}{\partial \mathbf{G}_p^{(j)}} = \mathcal{L}_{(\prod_{n<j} \mathbf{G}_p^{(n)})} \mathcal{R}_{(\prod_{n>j} \mathbf{G}_p^{(n)}) \mathbf{M}_{pq} (\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_q^{(n)})^H}, \tag{5.5}$$

and,

$$\frac{\partial \mathbf{V}_{pq}}{\partial \mathbf{G}_q^{(j)H}} = \mathcal{L}_{(\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_p^{(n)}) \mathbf{M}_{pq} (\prod_{n>j} \mathbf{G}_q^{(n)})^H} \mathcal{R}_{(\prod_{n<j} \mathbf{G}_q^{(n)})^H}. \tag{5.6}$$

These expressions should clarify the need for the operators described in the previous chapter. However, it should be noted that it is possible to express these derivatives as 4×4 matrices either via the isomorphism, $\mathbb{W}$, or by performing the differentiation on vectorised quantities. The 2×2 operators are substantially less cumbersome than the alternatives.

Returning to the formulation and definitions of Subsection 4.3.2 and noting that the variable of interest is the j'th Jones term, we can use the notation of Equation 4.67 to write down our augmented parameter, residual and model vectors as:

$$\breve{\mathbf{g}} = \begin{bmatrix} \vec{\mathbf{g}} \\ \vec{\mathbf{g}}^{\dagger} \end{bmatrix}, \qquad \breve{\mathbf{r}} = \begin{bmatrix} \vec{\mathbf{r}}(\breve{\mathbf{z}}) \\ \vec{\mathbf{r}}^{\dagger}(\breve{\mathbf{z}}) \end{bmatrix}, \qquad \breve{\mathbf{v}} = \begin{bmatrix} \vec{\mathbf{v}}(\breve{\mathbf{z}}) \\ \vec{\mathbf{v}}^{\dagger}(\breve{\mathbf{z}}) \end{bmatrix}, \tag{5.7}$$

where:

$$\vec{\mathbf{g}} = \left[\mathbf{G}_1^{(j)}, \ldots, \mathbf{G}_{N_\mathrm{A}}^{(j)}\right]^T, \quad \vec{\mathbf{r}} = \left[\mathbf{R}_{1\to 12}, \ldots, \mathbf{R}_{N_\mathrm{BL}\to pq}\right]^T, \quad \vec{\mathbf{v}} = \left[\mathbf{V}_{1\to 12}, \ldots, \mathbf{V}_{N_\mathrm{BL}\to pq}\right]^T. \tag{5.8}$$

The expressions for $\vec{\mathbf{r}}$ and $\vec{\mathbf{v}}$ emphasise the fact that each entry in the nested vector is associated with a specific baseline index pq which itself includes all combinations of antennas such that $p < q$.

Using these definitions, it is possible to write the Jacobian as:

$$\mathbf{J} = \frac{\partial \breve{\mathbf{v}}(\breve{\mathbf{g}})}{\partial \breve{\mathbf{g}}} = \begin{bmatrix} \frac{\partial \vec{\mathbf{v}}}{\partial \vec{\mathbf{g}}} & \frac{\partial \vec{\mathbf{v}}}{\partial \vec{\mathbf{g}}^{\dagger}} \\ \frac{\partial \vec{\mathbf{v}}^{\dagger}}{\partial \vec{\mathbf{g}}} & \frac{\partial \vec{\mathbf{v}}^{\dagger}}{\partial \vec{\mathbf{g}}^{\dagger}} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}, \tag{5.9}$$

where:

$$[\mathbf{A}]_{ab} = \frac{\partial \mathbf{V}_a}{\partial \mathbf{G}_b^{(j)}} = \mathcal{L}_{(\prod_{n<j} \mathbf{G}_p^{(n)})} \mathcal{R}_{(\prod_{n>j} \mathbf{G}_p^{(n)}) \mathbf{M}_{pq} (\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_q^{(n)})^H} \delta_p^b, \tag{5.10}$$

$$[\mathbf{B}]_{ab} = \frac{\partial \mathbf{V}_a}{\partial \mathbf{G}_b^{(j)H}} = \mathcal{L}_{(\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_p^{(n)}) \mathbf{M}_{pq} (\prod_{n>j} \mathbf{G}_q^{(n)})^H} \mathcal{R}_{(\prod_{n<j} \mathbf{G}_q^{(n)})^H} \delta_q^b, \tag{5.11}$$

$$[\mathbf{C}]_{ab} = \frac{\partial \mathbf{V}_a^H}{\partial \mathbf{G}_b^{(j)}} = \mathcal{L}_{(\prod_{n<j} \mathbf{G}_q^{(n)})} \mathcal{R}_{(\prod_{n>j} \mathbf{G}_q^{(n)}) \mathbf{M}_{pq}^H (\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_p^{(n)})^H} \delta_q^b, \tag{5.12}$$

$$[\mathbf{D}]_{ab} = \frac{\partial \mathbf{V}_a^H}{\partial \mathbf{G}_b^{(j)H}} = \mathcal{L}_{(\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_q^{(n)}) \mathbf{M}_{pq}^H (\prod_{n>j} \mathbf{G}_p^{(n)})^H} \mathcal{R}_{(\prod_{n<j} \mathbf{G}_p^{(n)})^H} \delta_p^b. \tag{5.13}$$

As can be seen above, the Jacobian has four $N_\mathrm{BL} \times N_\mathrm{A}$ blocks with entries consisting of 2×2 operators. The overall Jacobian is consequently a $2N_\mathrm{BL} \times 2N_\mathrm{A}$ matrix of operators. The entry at row a, column b of each block given by the expressions above. Again, each row in a block is associated with a specific baseline $(a \to pq)$ for all baselines such that $p < q$. The Dirac delta has it usual meaning and the Jacobian has relatively few non-zero entries. In fact, the exact percentage of non-zero entries is given by:

$$\text{percentage non-zero entries} = \frac{100}{N_\mathrm{A}}. \tag{5.14}$$

A close examination of the partial Jacobian expressions reveals that the vertically adjacent blocks produce the same expressions up to a conjugate transpose. This can be exploited, as was done in Smirnov and Tasse (2015), to write down a simpler expression for the Jacobian.

This is accomplished by modifying the pq index to include conjugate baselines (baselines for which $q < p$). Thus, pq will enumerate all combinations of p and q such that $p \neq q$. The resulting Jacobian consists of two $2N_{\text{BL}} \times N_{\text{A}}$ blocks:

$$\mathbf{J} = \frac{\partial \breve{\mathbf{v}}(\breve{\mathbf{g}})}{\partial \breve{\mathbf{g}}} = \begin{bmatrix} \dfrac{\partial \breve{\mathbf{v}}}{\partial \vec{\mathbf{g}}} & \dfrac{\partial \breve{\mathbf{v}}}{\partial \vec{\mathbf{g}}^{\dagger}} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix}, \tag{5.15}$$

where the definitions of $\mathbf{A}$ and $\mathbf{B}$ are unaltered.

The change to the indexing also allows us to simplify our definitions of the augmented model and residual vectors by noting that pq now includes the previously explicit conjugation:

$$\breve{\mathbf{r}} = \begin{bmatrix} \mathbf{R}_{1\to 12}, \ldots, \mathbf{R}_{2N_{\text{BL}}\to pq} \end{bmatrix}^T, \qquad \breve{\mathbf{v}} = \begin{bmatrix} \mathbf{V}_{1\to 12}, \ldots, \mathbf{V}_{2N_{\text{BL}}\to pq} \end{bmatrix}^T. \tag{5.16}$$

The order in which the elements appear in the augmented vectors can be arbitrary. However, we generally assume that all the non-conjugate baselines are enumerated before the conjugate baselines and that the non-conjugate and conjugate parts are ordered identically. This is best understood with an example. For the three antenna, three baseline case, the convention we follow would yield:

$$\begin{aligned} \breve{\mathbf{r}} &= \begin{bmatrix} \mathbf{R}_{12}, \mathbf{R}_{13}, \mathbf{R}_{23}, \mathbf{R}_{21}, \mathbf{R}_{31}, \mathbf{R}_{32} \end{bmatrix}^T \\ &= \begin{bmatrix} \mathbf{R}_{12}, \mathbf{R}_{13}, \mathbf{R}_{23}, \mathbf{R}_{12}^H, \mathbf{R}_{13}^H, \mathbf{R}_{23}^H \end{bmatrix}^T. \end{aligned} \tag{5.17}$$

5.1.3 Deriving an update rule

The Jacobian and the augmented residual are all that are required to make use of the Gauss-Newton and Levenberg-Marquardt algorithms. However, as in Subsection 4.2.2, there are several improvements and observations to be made which require further analysis of the problem.

Thus, we need an expression for $\mathbf{J}^H$, the conjugate transpose of the Jacobian. This can be obtained with relative ease from the expression for $\mathbf{J}$:

$$\mathbf{J}^H = \left(\frac{\partial \breve{\mathbf{v}}(\breve{\mathbf{g}})}{\partial \breve{\mathbf{g}}} \right)^H = \begin{bmatrix} \left(\dfrac{\partial \breve{\mathbf{v}}}{\partial \vec{\mathbf{g}}} \right)^H \\ \left(\dfrac{\partial \breve{\mathbf{v}}}{\partial \vec{\mathbf{g}}^{\dagger}} \right)^H \end{bmatrix} = \begin{bmatrix} \mathbf{A}^H \\ \mathbf{B}^H \end{bmatrix}, \tag{5.18}$$

where:

$$[\mathbf{A}^H]_{ab} = \mathcal{L}_{(\prod_{n<j} \mathbf{G}_p^{(n)})^H} \mathcal{R}_{(\prod_{n=1}^{N_{\text{J}}} \mathbf{G}_q^{(n)}) \mathbf{M}_{pq}^H (\prod_{n>j} \mathbf{G}_p^{(n)})^H} \delta_p^a, \tag{5.19}$$

$$[\mathbf{B}^H]_{ab} = \mathcal{L}_{(\prod_{n>j} \mathbf{G}_q^{(n)}) \mathbf{M}_{pq}^H (\prod_{n=1}^{N_{\text{J}}} \mathbf{G}_p^{(n)})^H} \mathcal{R}_{(\prod_{n<j} \mathbf{G}_q^{(n)})} \delta_q^a. \tag{5.20}$$

As in the case of the Jacobian, the rows of the individual blocks of the Jacobian are indexed by a and the columns by b. However, due to the conjugate transpose, it is the columns of $\mathbf{J}^H$ that are associated with specific baselines.

Armed with analytic expressions expressions for both $\mathbf{J}$ and $\mathbf{J}^H$, it is possible to write down the components of the Hessian. This is far more tractable than when we split the problem into its real and imaginary parts, largely due to the sparsity of the Jacobian. However, owing to the number of operators involved, it is first practical to define the following substitution:

$$\mathbf{Y}_{pq} = \Big(\prod_{n>j} \mathbf{G}_p^{(n)}\Big)\mathbf{M}_{pq}\Big(\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_q^{(n)}\Big)^H. \tag{5.21}$$

The approximation to the Hessian, $\mathbf{H}$, is then given by:

$$\mathbf{H} = \mathbf{J}^H\mathbf{J} = \begin{bmatrix}\mathbf{A}' & \mathbf{B}' \\ \mathbf{C}' & \mathbf{D}'\end{bmatrix}, \tag{5.22}$$

where:

$$[\mathbf{A}']_{ab} = \begin{cases}\mathcal{L}_{(\prod_{n<j}\mathbf{G}_a^{(n)})^H(\prod_{n<j}\mathbf{G}_a^{(n)})}\sum_{q\neq a}\mathcal{R}_{\mathbf{Y}_{aq}\mathbf{Y}_{aq}^H} & \text{if } a = b \\ 0 & \text{if } a \neq b\end{cases}, \tag{5.23}$$

$$[\mathbf{B}']_{ab} = \begin{cases}\mathcal{L}_{(\prod_{n<j}\mathbf{G}_a^{(n)})^H\mathbf{Y}_{ba}^H}\mathcal{R}_{(\prod_{n<j}\mathbf{G}_b^{(n)})^H\mathbf{Y}_{ab}^H} & \text{if } a \neq b \\ 0 & \text{if } a = b\end{cases}, \tag{5.24}$$

$$[\mathbf{C}']_{ab} = \begin{cases}\mathcal{R}_{\mathbf{Y}_{ba}(\prod_{n<j}\mathbf{G}_a^{(n)})}\mathcal{L}_{\mathbf{Y}_{ab}(\prod_{n<j}\mathbf{G}_b^{(n)})} & \text{if } a \neq b \\ 0 & \text{if } a = b\end{cases}, \tag{5.25}$$

$$[\mathbf{D}']_{ab} = \begin{cases}\mathcal{R}_{(\prod_{n<j}\mathbf{G}_a^{(n)})^H(\prod_{n<j}\mathbf{G}_a^{(n)})}\sum_{p\neq a}\mathcal{L}_{\mathbf{Y}_{ap}\mathbf{Y}_{ap}^H} & \text{if } a = b \\ 0 & \text{if } a \neq b\end{cases}. \tag{5.26}$$

Each block of the Hessian has shape $N_\mathrm{A} \times N_\mathrm{A}$ and contains the 2×2 operators given by the above expressions. The overall shape of the Hessian is $2N_\mathrm{A} \times 2N_\mathrm{A}$.

The final component of the update rule is the product of $\mathbf{J}^H$ and $\breve{\mathbf{r}}$:

$$\mathbf{J}^H\breve{\mathbf{r}} = \begin{bmatrix}\mathbf{E}' \\ \mathbf{F}'\end{bmatrix}, \tag{5.27}$$

where,

$$[\mathbf{E}']_a = \sum_{q\neq a}\Big(\prod_{n<j}\mathbf{G}_a^{(n)}\Big)^H\mathbf{R}_{aq}\mathbf{Y}_{aq}^H, \tag{5.28}$$

$$[\mathbf{F}']_a = \sum_{p\neq a}\mathbf{Y}_{ap}\mathbf{R}_{pa}\Big(\prod_{n<j}\mathbf{G}_a^{(n)}\Big). \tag{5.29}$$

The result of this product is a length $2N_\mathrm{A}$ vector of 2×2 matrices. It consists of two length N_A vectors, each indexed by a. It is also clear that the two halves of $\mathbf{J}^H\breve{\mathbf{r}}$ are Hermitian with respect to each other. Note the absence of operators in the expressions; they have already been applied to the residual term and what remains are relatively simple 2×2 matrix products.

We are thus armed with analytic expressions for all the components of the Gauss-Newton and Levenberg-Marquardt update rules for the case of a Jones chain. However, as was mentioned previously, there are some approximations and simplifications to be made.

The first such change is the based on the Hermitian nature of the problem. As half the equations are simply the conjugate transpose of the others, they do not actually add any new information. We are therefore free to discard half of each update. In the case of Gauss-Newton, the update rule then becomes:

$$\boldsymbol{\Delta}\vec{\mathbf{g}} = (\mathbf{J}^H\mathbf{J})_{\mathrm{U}}^{-1}\mathbf{J}^H\breve{\mathbf{r}}, \tag{5.30}$$

where $(\cdot)_{\mathrm{U}}$ denotes the upper half a of a matrix.

The following observation was made by Tasse (2014a):

$$\breve{\mathbf{v}} = \mathbf{J}_{\mathrm{L}}\vec{\mathbf{g}} = \frac{1}{2}\mathbf{J}\breve{\mathbf{g}}, \tag{5.31}$$

where $(\cdot)_{\mathrm{L}}$ denotes the left half of a matrix. Using this property and substituting for $\breve{\mathbf{r}}$ in equation 5.30, we obtain:

$$\boldsymbol{\Delta}\vec{\mathbf{g}} = (\mathbf{J}^H\mathbf{J})_{\mathrm{U}}^{-1}\mathbf{J}^H(\breve{\mathbf{d}} - \mathbf{J}_{\mathrm{L}}\vec{\mathbf{g}}) = (\mathbf{J}^H\mathbf{J})_{\mathrm{U}}^{-1}\mathbf{J}^H\breve{\mathbf{d}} - \vec{\mathbf{g}}, \tag{5.32}$$

where $\breve{\mathbf{d}}$ is the augmented data vector which is defined in the same way as the augmented residual vector using only the observed data.

If we introduce iteration index k in the above expression, and observe that $\vec{\mathbf{g}}_k + \boldsymbol{\Delta}\vec{\mathbf{g}} = \vec{\mathbf{g}}_{k+1}$, we can express the update in terms of the observed data rather than the residual:

$$\vec{\mathbf{g}}_{k+1} = (\mathbf{J}^H\mathbf{J})_{\mathrm{U}}^{-1}\mathbf{J}^H\breve{\mathbf{d}}. \tag{5.33}$$

A secondary effect of this change is that the update rule yields a vector of updated gain values directly, instead of a vector of gain updates. Note that although the residual has been replaced by the observed data, this expression is not model independent. Model terms still appear in the Jacobian.

Computing the residual is a computationally intensive task and avoiding it comes with a commensurate increase in performance. However, as was mentioned in Smirnov and Tasse (2015), the short-cut used above assumes an exact inversion of $\mathbf{H}$. If we approximate the Hessian in any way (with, for example, $\tilde{\mathbf{H}}$), the following must be satisfied to make use of Equation 5.32:

$$\tilde{\mathbf{H}}_{\mathrm{U}}^{-1}\mathbf{H}_{\mathrm{L}} = \mathbf{I}. \tag{5.34}$$

This is trivially true when $\tilde{\mathbf{H}}$ is an exact inversion of $\mathbf{H}$. If, however, Equation 5.34 is not satisfied it may be impossible to avoid the costly residual computation.

A discussion of the limitations of approximating the Hessian would be pointless unless we aimed to do just that. Returning to Equation 5.22, the summations in blocks $\mathbf{A}'$ and $\mathbf{D}'$ suggest that the diagonal entries of the Hessian are quite dominant. This is indeed the case

and we approximate our Hessian with a diagonal matrix given by:

$$\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{A}' & \mathbf{0} \\ \mathbf{0} & \mathbf{D}' \end{bmatrix}, \tag{5.35}$$

where the definitions of $\mathbf{A}'$ and $\mathbf{D}'$ are unchanged. Consequently, the diagonal entries remain 2×2 matrices. Inverting a diagonal matrix is far less computationally demanding than the alternatives ($\mathcal{O}(n)$ as opposed to $\mathcal{O}(n^3)$), and constitutes a large reduction in computational complexity. Additionally, this approximation still meets the requirement of Equation 5.34 and we are free to combine it with our other shortcuts. Before we do so, some mention must be made of the repercussions of this approximation.

An approximation is, by its nature, never quite as accurate as using the value which it approximates. However, in this instance, what is lost in iteration-to-iteration accuracy (which in turn affects the number of iterations taken to converge) is massively outweighed by the reduction in computational complexity. In other words, even if we take twice as many (or more) iterations to converge using the approximation, those iterations are so much faster than performing the full inversion that it is easy to justify.

Combining the approximation of the Hessian with Equation 5.33, we obtain:

$$\vec{\mathbf{g}}_{k+1} = \tilde{\mathbf{H}}_{\mathrm{UL}}^{-1} \mathbf{J}_{\mathrm{L}}^{H} \breve{\mathbf{d}}, \tag{5.36}$$

where $(\cdot)_{\mathrm{UL}}$ denotes the upper left quadrant of its associated matrix. This quadrant is ultimately the only part of the Hessian which is required due to the fact that Equation 5.33 excluded the lower half of the Hessian, and the diagonal approximation resulted in the $\mathbf{B}'$ block being zero everywhere.

Equation 5.36 has simplified the update rule substantially, and we can in fact already substitute the relevant blocks of $\mathbf{J}^H \breve{\mathbf{r}}$ and $\tilde{\mathbf{H}}$ directly into that expression to obtain:

$$\vec{\mathbf{g}}_{k+1} = (\mathbf{A}')^{-1} \mathbf{E}'. \tag{5.37}$$

We will now abandon the vector and matrix form above in favour of writing down a per-antenna update rule. This is very simple due to the diagonal structure of the $\mathbf{A}'$ block (there is no need to consider interactions between antennas) and all we need do is apply the relevant operators to the entries of $\mathbf{E}'$. Consequently, the update rule for the a'th antenna's j'th Jones term is:

$$\mathbf{G}_{a,k+1}^{(j)} = \left(\prod_{n<j} \mathbf{G}_{a}^{(n)} \right)^{-1} \left(\sum_{q \neq a} \mathbf{D}_{aq} \mathbf{Y}_{aq}^{H} \right) \left(\sum_{q \neq a} \mathbf{Y}_{aq} \mathbf{Y}_{aq}^{H} \right)^{-1}. \tag{5.38}$$

This remarkably simple expression can be used to update any term in a Jones chain and is one of the principle results of this work. Unlike more conventional approaches, it allows for several Jones terms to be determined without applying inverse terms to the data. This is, strictly speaking, the correct approach as it means that the data doesn't change as different terms in the chain are computed. It is worth noting that although the summations above are

defined over all baselines (excluding auto-correlations) we are free to omit any flagged terms from the summations. This is almost always necessary when dealing with real data due to RFI.

For a single Jones term, Equation 5.38 reduces to:

$$\mathbf{G}_{a,k+1} = \left(\sum_{q\neq a} \mathbf{D}_{aq}\mathbf{Y}_{aq}^{H}\right)\left(\sum_{q\neq a} \mathbf{Y}_{aq}\mathbf{Y}_{aq}^{H}\right)^{-1}, \tag{5.39}$$

with:

$$\mathbf{Y}_{aq} = \mathbf{M}_{aq}\mathbf{G}_{q}^{H}. \tag{5.40}$$

This update rule should agree with Equation 5.26 of Smirnov and Tasse (2015). However, due to an incorrectly applied operator in that paper, there is a discrepancy in the order of the terms. Regardless, as mentioned by Smirnov and Tasse (ibid.), this update rule is equivalent to polarised StEFCal (Salvini and Wijnholds 2014). This equivalence once again demonstrates how closely linked these calibration methods are.

One final point worth mentioning is that the update no longer looks particularly like the Gauss-Newton method we started with. Consequently, it is challenging to see how we could incorporate the improvements of the Levenberg-Marquardt algorithm. The critical insight here is that when approximating the Hessian by its diagonal, the Levenberg-Marquart trick of adding $\lambda\text{diag}(\mathbf{J}^H\mathbf{J})$ is essentially the same as multiplying the contents of the inverse term by λ. In practice, we do not often use λ explicitly and instead find the same empirical improvement to convergence when using the StEFCal strategy of averaging every even iteration with the preceding odd iteration.

5.1.4 Extending the update rule - solution intervals

The update rule presented in the preceding section deals only with the simplest of cases i.e. solving for an independent gain (Jones term) for each time and frequency point. This is often impossible due to insufficient signal-to-noise in the data and the danger of over-fitting (a proliferation of degrees-of-freedom).

Traditionally, this problem has been dealt with by using multiple time and/or frequency samples (denoted by the index s) in the computation of a single gain. This introduces the following modified form of the minimisation problem:

$$\min_{\{\mathbf{G}_p^{(n)}\}} \sum_{pqs} \|\mathbf{R}_{pqs}\|_F, \qquad \text{where} \qquad \mathbf{R}_{pqs} = \mathbf{D}_{pqs} - \mathbf{V}_{pqs}, \tag{5.41}$$

and,

$$\mathbf{V}_{pqs} = \left(\prod_{n=1}^{N_\text{J}} \mathbf{G}_p^{(n)}\right)\mathbf{M}_{pqs}\left(\prod_{n=1}^{N_\text{J}} \mathbf{G}_q^{(n)}\right)^{H}. \tag{5.42}$$

Redoing the derivation is impractical. Instead, we will draw attention to the changes that this formulation would require in the original derivation. The addition of time and/or frequency samples introduces additional equations which in turn make the Jacobian larger.

Specifically, the shape of the Jacobian becomes $2N_{\text{BL}}N_{\text{S}} \times 2N_{\text{A}}$, where N_{S} is the number of samples per gain.

Practically, this simply replaces every sum over baselines ($\sum_{p\neq q}$) with a sum over both baselines and samples ($\sum_{p\neq q,s}$). Making this change, the per antenna update rule becomes:

$$\mathbf{G}_{a,k+1}^{(j)} = \left(\prod_{n<j}\mathbf{G}_a^{(n)}\right)^{-1}\left(\sum_{q\neq a,s}\mathbf{D}_{aqs}\mathbf{Y}_{aqs}^H\right)\left(\sum_{q\neq a,s}\mathbf{Y}_{aqs}\mathbf{Y}_{aqs}^H\right)^{-1}, \tag{5.43}$$

where,

$$\mathbf{Y}_{pqs} = \left(\prod_{n>j}\mathbf{G}_p^{(n)}\right)\mathbf{M}_{pqs}\left(\prod_{n=1}^{N_{\text{J}}}\mathbf{G}_q^{(n)}\right)^H. \tag{5.44}$$

There is no reason, in the context of a Jones chain, for every term to share the same solution interval. In fact, using different solution intervals per term is crucial in allowing different terms to capture different behaviours. For example, a bandpass term needs high resolution in frequency but very low resolution in time whereas a generic **G** term is often chosen to have precisely the opposite properties. Provided we correctly broadcast the Jones terms against each other, the derived update rule allows us to use independent solution intervals per term.

5.1.5 Extending the update rule - weighted least squares

Weights are fairly ubiquitous in the field of interferometry. Their purpose can range from simply down-weighing poor data to adjusting the relative importance of different spatial scales. Naturally, this means that calibration is often formulated as a weighted least squares problem:

$$\min_{\{\mathbf{G}_p^{(n)}\}}\sum_{pqs}\|\mathbf{W}_{pqs}\odot\mathbf{R}_{pqs}\|_F, \qquad \text{where} \qquad \mathbf{R}_{pqs} = \mathbf{D}_{pqs} - \mathbf{V}_{pqs}. \tag{5.45}$$

Unfortunately, incorporating per-correlation weights into the Jones chain update rule is problematic - the introduction of a Hadamard product ($\odot$) makes the expressions difficult to manipulate. Consequently, we circumvent the problem by assuming that our weights are scalar and that the minimisation problem takes the following form:

$$\min_{\{\mathbf{G}_p^{(n)}\}}\sum_{pqs}\|w_{pqs}\mathbf{R}_{pqs}\|_F. \tag{5.46}$$

As scalar terms commute, this change makes it substantially easier to include the weights as they need only be pre-multiplied into both the observed visibilities (**D**) and the predicted visibilities (**M**, which appear in **V**).

5.1.6 Extending the update rule - direction dependence

As new interferometers are constructed and existing ones are upgraded, direction-dependent effects are becoming an increasingly limiting factor to the quality of their data products. Direction-dependent calibration is therefore of similarly increasing importance. However,

direction-dependent gain calibration has always been a computationally demanding task due to its reliance on N_{D} (number of directions) sets of predicted visibilities.

It is possible to extend our derivation to solve for N_{A} gains in each of N_{D} directions (indexed by d) by considering the following minimisation problem:

$$\min_{\{\mathbf{G}_{d,p}^{(n)}\}} \sum_{pqs} \|\mathbf{R}_{pqs}\|_F, \qquad \text{where} \qquad \mathbf{R}_{pqs} = \mathbf{D}_{pqs} - \sum_{d=1}^{N_{\mathrm{D}}} \mathbf{V}_{d,pqs}, \tag{5.47}$$

and,

$$\mathbf{V}_{d,pqs} = \left(\prod_{n=1}^{N_{\mathrm{J}}} \mathbf{G}_{d,p}^{(n)} \right) \mathbf{M}_{d,pqs} \left(\prod_{n=1}^{N_{\mathrm{J}}} \mathbf{G}_{d,q}^{(n)} \right)^H. \tag{5.48}$$

Smirnov and Tasse (2015) presented several strategies for solving the above minimisation problem in the context of complex optimisation. These strategies were centred around the degree to which the Hessian is approximated. Smirnov and Tasse (ibid.) focused on one of those methods, termed *CohJones*. CohJones is based on the assumption that the problem is separable by antenna but not by direction. As such, the resulting approximation of the Hessian has $N_{\mathrm{D}} \times N_{\mathrm{D}}$ blocks on its diagonal. Whilst a block matrix is cheaper to invert that an ordinary matrix, it can still become rather expensive as N_{D} grows.

We will focus on another of the presented strategies - *AllJones*. AllJones makes the greatest approximation by assuming that the problem is separable by both direction and antenna i.e. that, as in the direction-independent case, the Hessian can be approximated by a diagonal matrix with 2×2 entries. This strategy boasts the greatest reduction in computational complexity.

It is not unreasonable to expect the problem to be separable by direction provided that the directions we solve in are sufficiently far apart that the covariance between the directions can be neglected. This is the implicit assumption made by using only the diagonal of the Hessian. This is well-aligned with what happens in practice - several sources in close proximity are usually treated as being subject to the same direction-dependent gain.

Unfortunately, whilst AllJones has the greatest potential for accelerating the computation of the inverse of the Hessian, the diagonal approximation does not satisfy the requirement of Equation 5.34 and we are forced to revert to using the residuals rather than the observed visibilities. However, although using the residuals is more costly, it is nowhere near as expensive as computing a true inverse. It should be noted that the CohJones approximation does satisfy Equation 5.34.

Returning to the update rule for the direction-dependent case, we again omit an in-depth derivation and instead note how the addition of directions changes the existing one. Each additional direction brings with it an additional $2N_{\mathrm{A}}$ unknowns - the N_{A} gains associated with that direction and their conjugates. These additional unknowns in turn change the dimensions of the Jacobian. In the presence of both time/frequency samples and direction-dependent terms, the Jacobian will have shape $2N_{\mathrm{BL}}N_{\mathrm{S}} \times 2N_{\mathrm{A}}N_{\mathrm{D}}$. However, due to the diagonal approximation, the only major change to the update rule beyond the addition of a direction index and a reversion to the residuals will be that it becomes a per-antenna,

per-direction update rule:

$$\mathbf{G}^{(j)}_{d,a,k+1} = \left(\prod_{n<j} \mathbf{G}^{(n)}_{d,a} \right)^{-1} \left(\sum_{q \neq a,s} \mathbf{R}_{d,aqs} \mathbf{Y}^{H}_{d,aqs} \right) \left(\sum_{q \neq a,s} \mathbf{Y}_{d,aqs} \mathbf{Y}^{H}_{d,aqs} \right)^{-1}, \tag{5.49}$$

where:

$$\mathbf{Y}_{d,aqs} = \left(\prod_{n>j} \mathbf{G}^{(n)}_{d,a} \right) \mathbf{M}_{d,aqs} \left(\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}^{(n)}_{d,q} \right)^{H}. \tag{5.50}$$

The above is almost equivalent to solving N_D independent minimisation problems simultaneously. However, the different directions still communicate in some way via the residual.

Note that we have continued to use $\mathbf{G}$ to represent the Jones terms, even though we are discussing direction-dependent calibration for which $\mathbf{E}$ is the convention. This is to stress that the Jones terms are completely generic and may or may not be direction-dependent.

On that note, the above expression implies that every term in the chain must be direction-dependent. This is not the case. Provided the direction-dependent terms appear closer to the centre of the expression (closer to the "sky" term), we are free to include any combination of direction-independent and direction-dependent terms. If the current term of interest is direction-independent, it is necessary to sum over directions after applying all the direction-dependent terms. The converse is also true; if the current term of interest is direction-dependent, the direction-independent terms must be applied in all directions.

5.1.7 Calibration strategies

Whilst we briefly mentioned that we only solve for a single term in the chain at a time, we have not touched on how we move through the chain in the presence of multiple terms. There are two obvious options: solve to convergence for a single term before moving onto the next, or alternate between terms every few iterations.

The first option is often the best from a practical perspective - the inclusion of solved terms in the update rule means that the quality of data that each terms "sees" improves as we move through the chain. Additionally, as in the case of peeling, this strategy may be augmented by stepping through the chain several times. As all the terms are present in the update rules, there may be some room for improvement as more and more terms converge.

The second approach, alternating between terms every few iterations, may seem like the superior choice. We would expect that this approach would encourage each term to capture only the errors which the term is most suited to. In practice we have found that this approach can introduce problems with convergence and raises questions regarding the correct number of iterations to perform per term.

5.2 Specialised solvers

The preceding section presented a powerful method for calibrating a chain of generic Jones terms (gains). However, it is often preferable to specialise a solver in the interest of reducing

degrees-of-freedom and capturing specific behaviour in the solutions. This often allows us to extract more physical information than would be evident in the unspecialised case.

5.2.1 The chain rule

In order to construct specialised solvers, we need to develop methods for treating problems where the complex-valued variables are themselves functions of real parameters. Essentially, we need to solve the general problem of the following form:

$$\min_{\vec{\mathbf{x}}} \|\breve{\mathbf{d}} - \breve{\mathbf{v}}(\breve{\mathbf{z}}(\vec{\mathbf{x}}))\|_F, \tag{5.51}$$

where $\vec{\mathbf{x}}$ is a nested vector of real-valued parameters and all other symbols retain their previous definitions. Note that this describes both the scalar and 2×2 problem, depending on how the augmented and nested vectors are constructed. It is also worth noting that the parameter vector does not need to be augmented as it is real-valued.

In this case, the Jacobian of the problem will be given by the derivatives of the model term ($\breve{\mathbf{v}}$) with respect to the real-valued parameters ($\vec{\mathbf{x}}$). However, the chain rule of conventional calculus can be of use here. Instead of forming up the final Jacobian directly, we instead express it as the product of two Jacobians:

$$\mathbf{J} = \mathbf{J}_1\mathbf{J}_2, \quad \mathbf{J}_1 = \frac{\partial \breve{\mathbf{v}}}{\partial \breve{\mathbf{z}}}, \quad \mathbf{J}_2 = \frac{\partial \breve{\mathbf{z}}}{\partial \vec{\mathbf{x}}}. \tag{5.52}$$

The first Jacobian is already familiar to us, as it is identical to that used in the derivation of the Jones chain update rule. The second Jacobian is new and contains the partial derivatives of the complex variables with respect to the parameters. We will exploit this property in the derivations which follow.

5.2.2 A phase-only solver

One of the simplest to derive and yet surprisingly useful specialised solvers is a phase-only solver. A phase-only gain has unity amplitude but variable phase. Phase-only gain solutions are often used in the initial stages of data reduction pipelines when it is still difficult to constrain an amplitude solution. Applying a phase-only solution is more conservative and often makes it possible to build up an adequate model for subsequent amplitude and phase calibration.

The following derivation will make use of the 2×2 formulation but we will only consider diagonal gains i.e. we will assume that the cross correlation components of the gain terms are zero. This assumption is often made when performing phase-only calibration. We will also consider only a single Jones term. There are some limitations which prevent us from simply including specialised Jones terms in a chain using the 2×2 formalism. It is possible using a more general 4×4 matrix approach, but that has yet to be fully investigated.

Again, we restate the minimisation problem, omitting both solution intervals and directions for the time being:

$$\min_{\{\boldsymbol{\phi}_p\}} \sum_{pq} \|\mathbf{R}_{pq}\|_F, \qquad \text{where} \qquad \mathbf{R}_{pq} = \mathbf{D}_{pq} - \mathbf{V}_{pq}, \tag{5.53}$$

and,

$$\mathbf{V}_{pq} = \left(\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_p^{(n)} \right) \mathbf{M}_{pq} \left(\prod_{n=1}^{N_\mathrm{J}} \mathbf{G}_q^{(n)} \right)^H, \tag{5.54}$$

with,

$$\mathbf{G}_p = \begin{bmatrix} g_p^{XX} & 0 \\ 0 & g_p^{YY} \end{bmatrix} = \begin{bmatrix} e^{i\phi_p^{XX}} & 0 \\ 0 & e^{i\phi_p^{YY}} \end{bmatrix}. \tag{5.55}$$

All the symbols in the above expressions retain their previous definitions. The real-valued phases are denoted by ϕ and the superscripts XX and YY indicate the correlation with which they are associated. Note that we have assumed linear feeds in this instance, but that the derivation would hold equally true for circular feeds (with on-diagonal terms denoted by LL and RR).

The first step in deriving the update rule is to determine expressions for the two Jacobians, $\mathbf{J}_1$ and $\mathbf{J}_2$. The first Jacobian was derived in Subsection 5.1.2 but we restate it here for the simpler case of a single Jones term:

$$\mathbf{J}_1 = \frac{\partial \breve{\mathbf{v}}(\breve{\mathbf{g}})}{\partial \breve{\mathbf{g}}} = \begin{bmatrix} \dfrac{\partial \breve{\mathbf{v}}}{\partial \vec{\mathbf{g}}} & \dfrac{\partial \breve{\mathbf{v}}}{\partial \vec{\mathbf{g}}^\dagger} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix}, \tag{5.56}$$

where,

$$[\mathbf{A}]_{ab} = \mathbb{W}\mathcal{R}_{\mathbf{M}_{pq}\mathbf{G}_q^H}\delta_p^b, \tag{5.57}$$

$$[\mathbf{B}]_{ab} = \mathbb{W}\mathcal{L}_{\mathbf{G}_p\mathbf{M}_{pq}}\delta_q^b. \tag{5.58}$$

The sudden appearance of the isomorphism $\mathbb{W}$ in the above expressions requires some explanation. It is not possible to take the product of $\mathbf{J}_1$ with $\mathbf{J}_2$ if we leave $\mathbf{J}_1$ in operator form. This is due to the fact that the entries of $\mathbf{J}_2$ involve differentiating a 2×2 gain. Performing this differentiation requires us to vectorise the gain as a 4×1 vector before differentiating with respect to the parameter vector. The result of this differentiation is not a 2×2 matrix and we cannot express it as an operator. Consequently, we revert to using the 4×4 matrix representation. Thus, the entries of $\mathbf{J}_1$ are in fact 4×4 matrices.

For the phase-only solver, the nested parameter vector is given by:

$$\vec{\boldsymbol{\phi}} = \left[\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{N_\mathrm{A}} \right]^T, \tag{5.59}$$

where its constituent vectors are given by:

$$\boldsymbol{\phi}_p = \left[\phi_p^{XX}, \phi_p^{YY} \right]^T. \tag{5.60}$$

Due to this parameterisation, we introduce the notion of a number parameters per antenna, and denote this quantity using N_{PPA}. Thus, $\mathbf{J}_2$ has dimensions $2N_{\text{A}} \times N_{\text{A}}N_{\text{PPA}}$ and is given by:

$$\mathbf{J}_2 = \frac{\partial \breve{\mathbf{g}}}{\partial \vec{\boldsymbol{\phi}}} = \begin{bmatrix} \mathbf{C} \\ \mathbf{D} \end{bmatrix}, \tag{5.61}$$

where:

$$[\mathbf{C}]_{ab} = \frac{\partial \mathbf{g}_p}{\partial \boldsymbol{\phi}_b} \delta_p^b, \tag{5.62}$$

$$[\mathbf{D}]_{ab} = \frac{\partial \mathbf{g}_q^\star}{\partial \boldsymbol{\phi}_b} \delta_q^b, \tag{5.63}$$

and it is understood that:

$$\mathbf{g}_p = \text{vec}(\mathbf{G}_p), \quad \mathbf{g}_q^\star = \text{vec}(\mathbf{G}_q^H). \tag{5.64}$$

The rows and columns of the $N_{\text{A}} \times N_{\text{A}}N_{\text{PPA}}$ blocks $\mathbf{C}$ and $\mathbf{D}$ are again indexed by a and b. Each entry in $\mathbf{J}_2$ is a 4×2 matrix and, as such, they are compatible with the 4×4 entries of $\mathbf{J}_1$. Each row in $\mathbf{C}$ and $\mathbf{D}$ is associated with a specific antenna - in fact, that antenna is given by a. However, we retain the p and q subscripts to maintain clarity in later parts of the derivation.

With the definitions above, we can combine $\mathbf{J}_1$ and $\mathbf{J}_2$ and write the overall Jacobian as:

$$\mathbf{J} = \mathbf{J}_1\mathbf{J}_2 = \Big[\mathbf{AC} + \mathbf{BD}\Big], \tag{5.65}$$

the entries of which are given by:

$$[\mathbf{J}]_{ab} = \mathbb{W}\mathcal{R}_{\mathbf{M}_{pq}\mathbf{G}_q^H} \frac{\partial \mathbf{g}_p}{\partial \boldsymbol{\phi}_b} \delta_p^b + \mathbb{W}\mathcal{L}_{\mathbf{G}_p\mathbf{M}_{pq}} \frac{\partial \mathbf{g}_q^\star}{\partial \boldsymbol{\phi}_b} \delta_q^b, \tag{5.66}$$

recalling that each row is associated with a specific baseline ($a \to pq$). The final Jacobian has shrunk to a size of $2N_{\text{BL}} \times N_{\text{A}}N_{\text{PPA}}$ and it does not have an obvious left and right half. This makes it even simpler to write down a similar expression for $\mathbf{J}^H$ (with dimensions of $N_{\text{A}}N_{\text{PPA}} \times 2N_{\text{BL}}$), the entries of which are given by:

$$[\mathbf{J}^H]_{ab} = \left(\frac{\partial \mathbf{g}_p}{\partial \boldsymbol{\phi}_a}\right)^H \mathbb{W}\mathcal{R}_{\mathbf{G}_q\mathbf{M}_{pq}^H} \delta_p^a + \left(\frac{\partial \mathbf{g}_q^\star}{\partial \boldsymbol{\phi}_a}\right)^H \mathbb{W}\mathcal{L}_{\mathbf{M}_{pq}^H\mathbf{G}_p^H} \delta_q^a. \tag{5.67}$$

A secondary consequence of this smaller combined Jacobian is that the dimensions of $\mathbf{J}^H\mathbf{J}$ also change, and the approximation of the Hessian has shape $N_{\text{A}}N_{\text{PPA}} \times N_{\text{A}}N_{\text{PPA}}$. Additionally, it no longer has quadrants and we can write down the following:

$$\mathbf{H} = \mathbf{J}^H\mathbf{J} = \Big[\mathbf{A}'\Big], \tag{5.68}$$

where,

$$[\mathbf{A}']_{ab} = \begin{cases} \sum_{q\neq a}\left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)^H \mathbb{W}\mathcal{R}_{\mathbf{M}_{aq}\mathbf{G}_q^H\mathbf{G}_q\mathbf{M}_{aq}^H}\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a} + \sum_{p\neq a}\left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a}\right)^H \mathbb{W}\mathcal{L}_{\mathbf{M}_{pa}^H\mathbf{G}_p^H\mathbf{G}_p\mathbf{M}_{pa}}\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a} & \text{if } a = b, \\ \left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)^H \mathbb{W}\mathcal{R}_{\mathbf{G}_b\mathbf{M}_{ab}^H}\mathbb{W}\mathcal{L}_{\mathbf{G}_a\mathbf{M}_{ab}}\frac{\partial \mathbf{g}_b^\star}{\partial \boldsymbol{\phi}_b} + \left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a}\right)^H \mathbb{W}\mathcal{L}_{\mathbf{M}_{ba}^H\mathbf{G}_b^H}\mathbb{W}\mathcal{R}_{\mathbf{M}_{ba}\mathbf{G}_a^H}\frac{\partial \mathbf{g}_b}{\partial \boldsymbol{\phi}_b} & \text{if } a \neq b. \end{cases} \tag{5.69}$$

The entries of the resulting $\mathbf{J}^H\mathbf{J}$ are $N_{\text{PPA}} \times N_{\text{PPA}}$ blocks, which just so happen to be 2×2 in this case (two phases per antenna). It is the multiplication by the gain-by-parameter derivative terms which introduces this change in the size of the entries.

The other component of the update rule is of course the $\mathbf{J}^H\check{\mathbf{r}}$ term. In the phase-only case, it is an $N_{\text{A}}N_{\text{PPA}} \times 1$ vector. However, here we treat it as an $N_{\text{A}} \times 1$ nested vector, the entries of which are $N_{\text{PPA}} \times 1$ vectors given by:

$$[\mathbf{J}^H\check{\mathbf{r}}]_a = \sum_{q\neq a}\left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)^H \mathbb{W}\mathcal{R}_{\mathbf{G}_q\mathbf{M}_{aq}^H}\,\text{vec}(\mathbf{R}_{aq}) + \sum_{p\neq a}\left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a}\right)^H \mathbb{W}\mathcal{L}_{\mathbf{M}_{pa}^H\mathbf{G}_p^H}\,\text{vec}(\mathbf{R}_{pa}). \tag{5.70}$$

The vectorisation of the residual terms is necessary due to the use of the 4×4 matrix form. Equation 5.70 is equivalent to the following:

$$[\mathbf{J}^H\check{\mathbf{r}}]_a = \sum_{q\neq a}\left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)^H \text{vec}(\mathbf{R}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H) + \sum_{p\neq a}\left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a}\right)^H \text{vec}(\mathbf{M}_{pa}^H\mathbf{G}_p^H\mathbf{R}_{pa}), \tag{5.71}$$

which follows from the definition of the isomorphism. In order to go any further with this expression, we need a more thorough understanding of the derivative terms. Those appearing in Equation 5.71 are given by:

$$\left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)^H = \begin{bmatrix} -i\bar{g}_a^{XX} & 0 & 0 & 0 \\ 0 & 0 & 0 & -i\bar{g}_a^{YY} \end{bmatrix}, \tag{5.72}$$

$$\left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a}\right)^H = \begin{bmatrix} ig_a^{XX} & 0 & 0 & 0 \\ 0 & 0 & 0 & ig_a^{YY} \end{bmatrix}. \tag{5.73}$$

As the second and third columns of the these expression contain only zeros, it is clear that only the first and last entries of the vectorised components in Equation 5.71 will appear in the product. We exploit this property to write down another equivalent expression for $\mathbf{J}^H\check{\mathbf{r}}$:

$$[\mathbf{J}^H\check{\mathbf{r}}]_a = \sum_{q\neq a}\text{diag}(-i\mathbf{G}_a^H \odot (\mathbf{R}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H)) + \sum_{p\neq a}\text{diag}(i\mathbf{G}_a \odot (\mathbf{M}_{pa}^H\mathbf{G}_p^H\mathbf{R}_{pa})), \tag{5.74}$$

where $\odot$ denotes a Hadamard (element-wise) product and $\text{diag}(\cdot)$ extracts the diagonal entries of its argument and stacks them as a vector. Equation 5.74 also reveals the fact that the second summation is the conjugate of the first. For any complex number z we know that

$z + \bar{z} = 2\,\mathbb{R}\mathrm{e}(z)$, noting that $2\,\mathbb{R}\mathrm{e}(\cdot)$ extracts the real part of its argument. Applying this to Equation 5.74 we obtain:

$$[\mathbf{J}^H \breve{\mathbf{r}}]_a = \sum_{q \neq a} \mathrm{diag}(2\,\mathbb{R}\mathrm{e}(-i\mathbf{G}_a^H \odot (\mathbf{R}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H))). \tag{5.75}$$

This is almost the end of our manipulation of the $\mathbf{J}^H\breve{\mathbf{r}}$ term. However, it is still possible to note that $\mathbb{R}\mathrm{e}(-iz) = \mathbb{I}\mathrm{m}(z)$ where $\mathbb{I}\mathrm{m}(\cdot)$ extracts the imaginary part of its argument. Thus:

$$[\mathbf{J}^H \breve{\mathbf{r}}]_a = \sum_{q \neq a} \mathrm{diag}(2\,\mathbb{I}\mathrm{m}(\mathbf{G}_a^H \odot (\mathbf{R}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H))). \tag{5.76}$$

Combining Equations 5.68 and 5.76 is sufficient for writing down an update rule. However, as in the case of the Jones chain, we first make some approximations and apply some shortcuts in order to improve the performance of the solver.

The first step is to approximate the $\mathbf{J}^H\mathbf{J}$ term by its $N_{\mathrm{PPA}} \times N_{\mathrm{PPA}}$ diagonal entries. This is simple, as it is just the $a = b$ component of Equation 5.68. However, it is possible to simplify that expression further by noting that in the special case of diagonal, phase-only (unity amplitude) gains:

$$\mathbf{G}_p^H\mathbf{G}_p = \mathbf{G}_p\mathbf{G}_p^H = \mathbf{I}. \tag{5.77}$$

Combining this simplification with the analytic forms of the gain-by-parameter derivative terms, we obtain the following hugely simplified expression for the entries of $\tilde{\mathbf{H}}$:

$$[\tilde{\mathbf{H}}]_a = 2\sum_{q \neq a} \mathbf{I} \odot (\mathbf{M}_{aq}\mathbf{M}_{aq}^H). \tag{5.78}$$

Equation 5.78 is a startling and remarkable result: the diagonal approximation of the $\mathbf{J}^H\mathbf{J}$ term is invariant with respect to the gains. This means that it does not vary from iteration to iteration and it is only necessary to compute and invert it once. Of course, this massively accelerates individual iterations which was the original goal of the approximation.

There is still one other shortcut to be employed before presenting the final update rule. Whilst the requirement of Equation 5.34 is not met, we can achieve a similar result simply by substituting $\mathbf{R}_{pq} = \mathbf{D}_{pq} - \mathbf{G}_p\mathbf{M}_{pq}\mathbf{G}_q^H$ in Equation 5.76. Doing so, it rapidly becomes clear that all terms not containing $\mathbf{D}_{pq}$ are zero and we can once again avoid using the residual. Thus, we are left with:

$$[\mathbf{J}^H \breve{\mathbf{r}}]_a = \sum_{q \neq a} \mathrm{diag}(2\,\mathbb{I}\mathrm{m}(\mathbf{G}_a^H \odot (\mathbf{D}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H))). \tag{5.79}$$

It is however critical to note that this is not identical to the shortcut taken in the derivation of the Jones chain. We do not end up computing the new gains directly - we still compute an update.

Finally, combining all the simplified expressions, we are left with the per-antenna, phase-only update rule:

$$\boldsymbol{\Delta\phi}_a = \left(\sum_{q\neq a} \mathbf{I} \odot (\mathbf{M}_{aq}\mathbf{M}_{aq}^H)\right)^{-1} \left(\sum_{q\neq a} \text{diag}(\mathbb{I}\text{m}(\mathbf{G}_a^H \odot \mathbf{D}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H))\right), \tag{5.80}$$

where the update to the a'th antenna is a vector of the N_{PPA} (two in this case) values corresponding to the change in its parameters. In this instance, the update is to the real-valued phases and it is necessary to compute a new value of $\mathbf{G}_a$ using the updated parameters. This update rule can be extended in the same fashion as the Jones chain.

5.2.3 An alternative approach to solver specialisation

We have shown that it is possible to derive an update rule for a parameterised Jones term. However, the derivation we employed was slightly arduous. It would be a more powerful result if we could write down an update rule for an arbitrary parameterisation. Doing so requires us to infer the parameterised update rule using what we already know about a general complex gain.

The first step is to return to the Gauss-Newton update rule and the parameterised case where $\mathbf{J} = \mathbf{J}_1\mathbf{J_2}$. Making this substitution, we find that the update to an arbitrary real-valued nested parameter vector, $\vec{\mathbf{x}}$ (length N_{A} vector the entries of which are N_{PPA} vectors), is given by:

$$\boldsymbol{\Delta}\vec{\mathbf{x}} = [\mathbf{J}^H\mathbf{J}]^{-1}\mathbf{J}^H\check{\mathbf{r}}(\check{\mathbf{z}}(\vec{\mathbf{x}})) = [\mathbf{J}_2^H\mathbf{J}_1^H\mathbf{J}_1\mathbf{J}_2]^{-1}\mathbf{J}_2^H\mathbf{J}_1^H\check{\mathbf{r}}(\check{\mathbf{z}}(\vec{\mathbf{x}})). \tag{5.81}$$

Grouping terms, we obtain:

$$\boldsymbol{\Delta}\vec{\mathbf{x}} = [\mathbf{J}_2^H(\mathbf{J}_1^H\mathbf{J}_1)\mathbf{J}_2]^{-1}\mathbf{J}_2^H(\mathbf{J}_1^H\check{\mathbf{r}}(\check{\mathbf{z}}(\vec{\mathbf{x}}))) = [\mathbf{J}_2^H\tilde{\mathbf{H}}\mathbf{J}_2]^{-1}\mathbf{J}_2^H(\mathbf{J}_1^H\check{\mathbf{r}}(\check{\mathbf{z}}(\vec{\mathbf{x}}))), \tag{5.82}$$

where we have replaced $\mathbf{J}_1^H\mathbf{J}_1$ with $\tilde{\mathbf{H}}$, the the diagonal approximation of the Hessian obtained in Subsection 5.1.3. We restate it here, both for the sake of clarity and to cast it into its single term, 4×4 matrix form:

$$\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{A}' & \mathbf{0} \\ \mathbf{0} & \mathbf{D}' \end{bmatrix}, \tag{5.83}$$

where:

$$[\mathbf{A}']_{ab} = \begin{cases} \sum_{q\neq a} \mathbb{W}\mathcal{R}_{\mathbf{M}_{aq}\mathbf{G}_q^H\mathbf{G}_q\mathbf{M}_{aq}^H} & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases}, \tag{5.84}$$

$$[\mathbf{D}']_{ab} = \begin{cases} \sum_{p\neq a} \mathbb{W}\mathcal{L}_{\mathbf{M}_{pa}^H\mathbf{G}_p^H\mathbf{G}_p\mathbf{M}_{pa}} & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases}. \tag{5.85}$$

The Jacobian associated with the the parametrisation, $\mathbf{J}_2$, is given by:

$$\mathbf{J}_2 = \frac{\partial\check{\mathbf{g}}}{\partial\vec{\mathbf{x}}} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}. \tag{5.86}$$

where:

$$[\mathbf{A}]_{ab} = \frac{\partial \mathbf{g}_p}{\partial \mathbf{x}_b}\delta_p^b, \tag{5.87}$$

$$[\mathbf{B}]_{ab} = \frac{\partial \mathbf{g}_q^\star}{\partial \mathbf{x}_b}\delta_q^b. \tag{5.88}$$

Its conjugate transpose is given by:

$$\mathbf{J}_2^H = \left(\frac{\partial \breve{\mathbf{g}}}{\partial \vec{\mathbf{x}}}\right)^H = \begin{bmatrix}\mathbf{A}^H & \mathbf{B}^H\end{bmatrix}, \tag{5.89}$$

where:

$$[\mathbf{A}^H]_{ab} = \left(\frac{\partial \mathbf{g}_p}{\partial \mathbf{x}_a}\right)^H \delta_p^a, \tag{5.90}$$

$$[\mathbf{B}^H]_{ab} = \left(\frac{\partial \mathbf{g}_q^\star}{\partial \mathbf{x}_a}\right)^H \delta_q^a. \tag{5.91}$$

Taking the product of $\mathbf{J}_2^H$, $\tilde{\mathbf{H}}$ and $\mathbf{J}_2$ we find that its diagonal entries are given by:

$$[\mathbf{J}_2^H \tilde{\mathbf{H}} \mathbf{J}_2]_{aa} = \sum_{q\neq a}\left(\frac{\partial \mathbf{g}_a}{\partial \mathbf{x}_a}\right)^H \mathbb{W}\mathcal{R}_{\mathbf{M}_{aq}\mathbf{G}_q^H\mathbf{G}_q\mathbf{M}_{aq}^H}\frac{\partial \mathbf{g}_a}{\partial \mathbf{x}_a} + \sum_{p\neq a}\left(\frac{\partial \mathbf{g}_a^\star}{\partial \mathbf{x}_a}\right)^H \mathbb{W}\mathcal{L}_{\mathbf{M}_{pa}^H\mathbf{G}_p^H\mathbf{G}_p\mathbf{M}_{pa}}\frac{\partial \mathbf{g}_a^\star}{\partial \mathbf{x}_a}. \tag{5.92}$$

The above expression should be very familiar. It is, in fact, completely identical to the diagonal component of Equation 5.68 if we choose $\vec{\boldsymbol{\phi}}$ as our parameter vector i.e. set $\vec{\mathbf{x}} = \vec{\boldsymbol{\phi}}$. This is a general expression for the diagonal entries of the overall Hessian when we assume that $\mathbf{J}_1^H\mathbf{J}_1$ is diagonal. It neatly expresses the fact that the diagonal entries of the $N_\mathrm{A} \times N_\mathrm{A}$ Hessian will be $N_\mathrm{PPA} \times N_\mathrm{PPA}$ blocks due to the dimensions of the derivative terms. Thus, regardless of the choice of parameterisation, we will end up with an easy-to-invert block diagonal matrix.

Having found an alternative formulation for the diagonal entries of the Hessian, we can do the same for the $\mathbf{J}_2^H\mathbf{J}_1^H\breve{\mathbf{r}}$ term. From Equation 5.27, we already have expressions for $\mathbf{J}_1^H\breve{\mathbf{r}}$. We rewrite them here, again using only a single Jones term and applying the necessary vectorisation:

$$\mathbf{J}^H\breve{\mathbf{r}} = \begin{bmatrix}\mathbf{E}' \\ \mathbf{F}'\end{bmatrix}, \tag{5.93}$$

where,

$$[\mathbf{E}']_a = \sum_{q\neq a}\mathrm{vec}(\mathbf{R}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H), \tag{5.94}$$

$$[\mathbf{F}']_a = \sum_{p\neq a}\mathrm{vec}(\mathbf{M}_{pa}^H\mathbf{G}_p^H\mathbf{R}_{pa}). \tag{5.95}$$

Multiplying in $\mathbf{J}_2^H$, we find that the entries of this nested length N_A vector are length N_PPA vectors given by:

$$[\mathbf{J}_2^H \mathbf{J}_1^H \breve{\mathbf{r}}]_a = \sum_{q \neq a} \left(\frac{\partial \mathbf{g}_a}{\partial \mathbf{x}_a} \right)^H \text{vec}(\mathbf{R}_{aq} \mathbf{G}_q \mathbf{M}_{aq}^H) + \sum_{p \neq a} \left(\frac{\partial \mathbf{g}_a^\star}{\partial \mathbf{x}_a} \right)^H \text{vec}(\mathbf{M}_{pa}^H \mathbf{G}_p^H \mathbf{R}_{pa}). \tag{5.96}$$

Once again, this expression is equivalent to Equation 5.71 provided we choose $\vec{\mathbf{x}} = \vec{\boldsymbol{\phi}}$. There is, however, one additional observation to be made using Equation 5.96 - regardless of the parameterisation, its two constituent terms are always conjugates of each other. Thus, we can further simplify the general expression and write:

$$[\mathbf{J}_2^H \mathbf{J}_1^H \breve{\mathbf{r}}]_a = \sum_{q \neq a} 2\mathbb{R}\text{e}\left(\left(\frac{\partial \mathbf{g}_a}{\partial \mathbf{x}_a} \right)^H \text{vec}(\mathbf{R}_{aq} \mathbf{G}_q \mathbf{M}_{aq}^H) \right). \tag{5.97}$$

5.2.4 A phase-slope solver

It is possible to derive other, potentially more powerful and interesting phase-only solvers by considering alternative and perhaps more explicit parametrisations of the gain. An example of this would be a solver specialised for the purpose of calibrating delay errors. Delay errors manifest as a linear slope in phase across frequency. Thus, a delay solver would need to determine the parameters of that slope. This amounts to finding the value of just two parameters per antenna - a gradient and offset - over a large bandwidth.

Delay calibration and its twin, rate calibration (similar, but with slopes across time) are most commonly used in the field of VLBI (see Cotton (1995) and the references therein). We will make use of the alternative approach elaborated upon in the previous section in order to derive our own delay and rate solvers.

The derivation to follow will consider the general case of independent slopes across both time and frequency with a single intercept. It will be simple to modify the resulting update rule for the case when we consider only one of the slopes. Like the previous derivations, we will omit solution intervals in the interest of keeping the notation simple. However, they are a critical component of this solver due to the fact that it is impossible to constrain a slope and intercept using only a single data point. They will be reintroduced at the end of the derivation.

The minimisation problem we intend to solve is given by:

$$\min_{\{\gamma_p\}} \sum_{pq} \|\mathbf{R}_{pq}\|_F, \qquad \text{where} \qquad \mathbf{R}_{pq} = \mathbf{D}_{pq} - \mathbf{V}_{pq}, \tag{5.98}$$

and,

$$\mathbf{V}_{pq} = \left(\prod_{n=1}^{N_\text{J}} \mathbf{G}_p^{(n)} \right) \mathbf{M}_{pq} \left(\prod_{n=1}^{N_\text{J}} \mathbf{G}_q^{(n)} \right)^H, \tag{5.99}$$

with,

$$\mathbf{G}_p = \begin{bmatrix} g_p^{XX} & 0 \\ 0 & g_p^{YY} \end{bmatrix} = \begin{bmatrix} e^{i(a_p^{XX}\nu + b_p^{XX} t + c_p^{XX})} & 0 \\ 0 & e^{i(a_p^{YY}\nu + b_p^{YY} t + c_p^{YY})} \end{bmatrix}. \tag{5.100}$$

In the above expressions, ν is the frequency associated with the visibility, t is the time at which it was acquired and the coefficients a, b and c are the gradient in phase, gradient in time and intercept respectively. Note that the coefficients differ per correlation. We also draw attention to the fact that the coefficients should not be confused with the row and column indices a and b. They are usually easily differentiated by context.

The parameter vector, $\vec{\boldsymbol{\gamma}}$, with respect to which we perform the minimisation has not yet been defined. It is given by the following (technically double) nested vector:

$$\vec{\boldsymbol{\gamma}} = \left[\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_{N_\mathrm{A}}\right]^T = \left[\mathbf{a}_1, \mathbf{b}_1, \mathbf{c}_1, \ldots, \mathbf{a}_{N_\mathrm{A}}, \mathbf{b}_{N_\mathrm{A}}, \mathbf{c}_{N_\mathrm{A}}\right]^T, \tag{5.101}$$

with:

$$\mathbf{a}_p = \left[a_p^{XX}, a_p^{YY}\right]^T, \qquad \mathbf{b}_p = \left[b_p^{XX}, b_p^{YY}\right]^T, \qquad \mathbf{c}_p = \left[c_p^{XX}, c_p^{YY}\right]^T. \tag{5.102}$$

We now return to Equations 5.92 and 5.97 and replace the arbitrary parameter vector with $\vec{\boldsymbol{\gamma}}$. This yields:

$$[\mathbf{J}_2^H \tilde{\mathbf{H}} \mathbf{J}_2]_{aa} = \sum_{q \neq a} \left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\gamma}_a}\right)^H \mathbb{W}\mathcal{R}_{\mathbf{M}_{aq}\mathbf{G}_q^H\mathbf{G}_q\mathbf{M}_{aq}^H} \frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\gamma}_a} + \sum_{p \neq a} \left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\gamma}_a}\right)^H \mathbb{W}\mathcal{L}_{\mathbf{M}_{pa}^H\mathbf{G}_p^H\mathbf{G}_p\mathbf{M}_{pa}} \frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\gamma}_a}, \tag{5.103}$$

and,

$$[\mathbf{J}_2^H \mathbf{J}_1^H \check{\mathbf{r}}]_a = \sum_{q \neq a} 2\mathbb{R}\mathrm{e}\left(\left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\gamma}_a}\right)^H \mathrm{vec}(\mathbf{R}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H)\right). \tag{5.104}$$

As usual, we want to further simplify these expressions. To do so, we once again need some additional understanding of the derivative terms. In this case there are six parameters per antenna and the vector-by-vector derivatives yield $4 \times N_\mathrm{PPA}$ matrices, the conjugate transposes of which are given by:

$$\left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\gamma}_a}\right)^H = \begin{bmatrix} \left(\frac{\partial \mathbf{g}_a}{\partial \mathbf{a}_a}\right)^H \\ \left(\frac{\partial \mathbf{g}_a}{\partial \mathbf{b}_a}\right)^H \\ \left(\frac{\partial \mathbf{g}_a}{\partial \mathbf{c}_a}\right)^H \end{bmatrix} = \begin{bmatrix} -i\nu\bar{g}_a^{XX} & 0 & 0 & 0 \\ 0 & 0 & 0 & -i\nu\bar{g}_a^{YY} \\ -it\bar{g}_a^{XX} & 0 & 0 & 0 \\ 0 & 0 & 0 & -it\bar{g}_a^{YY} \\ -i\bar{g}_a^{XX} & 0 & 0 & 0 \\ 0 & 0 & 0 & -i\bar{g}_a^{YY} \end{bmatrix}, \tag{5.105}$$

$$\left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\gamma}_a}\right)^H = \begin{bmatrix} \left(\frac{\partial \mathbf{g}_a^\star}{\partial \mathbf{a}_a}\right)^H \\ \left(\frac{\partial \mathbf{g}_a^\star}{\partial \mathbf{b}_a}\right)^H \\ \left(\frac{\partial \mathbf{g}_a^\star}{\partial \mathbf{c}_a}\right)^H \end{bmatrix} = \begin{bmatrix} i\nu g_a^{XX} & 0 & 0 & 0 \\ 0 & 0 & 0 & i\nu g_a^{YY} \\ itg_a^{XX} & 0 & 0 & 0 \\ 0 & 0 & 0 & itg_a^{YY} \\ ig_a^{XX} & 0 & 0 & 0 \\ 0 & 0 & 0 & ig_a^{YY} \end{bmatrix}. \tag{5.106}$$

It is striking how similar the three blocks in each derivative are. Each block differs from the others by a single multiplicative factor. The factor is the coefficient of the parameter

with respect to which the derivative is taken. In the case of the intercept, that coefficient is one and the derivative is identical to the phase-only case. We exploit this insight into the structure of the derivative terms to write out the following alternative expressions in terms of the Kronecker product ($\otimes$):

$$\left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\gamma}_a}\right)^H = \mathbf{f} \otimes \left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)^H, \tag{5.107}$$

$$\left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\gamma}_a}\right)^H = \mathbf{f} \otimes \left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a}\right)^H, \tag{5.108}$$

where,

$$\mathbf{f} = \begin{bmatrix} \nu \\ t \\ 1 \end{bmatrix}. \tag{5.109}$$

Substituting these new derivative expressions back into Equation 5.103 and 5.104 we obtain:

$$\begin{aligned}[\mathbf{J}_2^H \tilde{\mathbf{H}} \mathbf{J}_2]_{aa} = &\sum_{q \neq a} \left(\mathbf{f} \otimes \left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)^H\right) \mathbb{W}\mathcal{R}_{\mathbf{M}_{aq}\mathbf{G}_q^H\mathbf{G}_q\mathbf{M}_{aq}^H}\left(\mathbf{f}^T \otimes \left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)\right) \; + \\ &\sum_{p \neq a} \left(\mathbf{f} \otimes \left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a}\right)^H\right) \mathbb{W}\mathcal{L}_{\mathbf{M}_{pa}^H\mathbf{G}_p^H\mathbf{G}_p\mathbf{M}_{pa}}\left(\mathbf{f}^T \otimes \left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a}\right)\right),\end{aligned} \tag{5.110}$$

and:

$$[\mathbf{J}_2^H \mathbf{J}_1^H \breve{\mathbf{r}}]_a = \sum_{q \neq a} 2\mathbb{R}\mathrm{e}\left(\left(\mathbf{f} \otimes \left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)^H\right) \mathrm{vec}(\mathbf{R}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H)\right). \tag{5.111}$$

At first it may appear that all we have succeeded in doing is making the expressions even longer and more cumbersome to write down. However, the value of this substitution becomes apparent when we apply the mixed-product property of Kronecker products. For matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ such that it is possible to form the matrix products $\mathbf{AC}$ and $\mathbf{BD}$, the mixed-product property states:

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}). \tag{5.112}$$

This property becomes even more useful when we note that we can express any matrix $\mathbf{A}$ as a Kronecker product with $\mathbf{1}$. In other words, given that $\mathbf{1}$ is a 1×1 matrix containing a single one, $\mathbf{A} \equiv \mathbf{A} \otimes \mathbf{1} \equiv \mathbf{1} \otimes \mathbf{A}$. We use this to simplify Equation 5.111:

$$\begin{aligned}[\mathbf{J}_2^H \mathbf{J}_1^H \breve{\mathbf{r}}]_a &= \sum_{q \neq a} 2\mathbb{R}\mathrm{e}\left(\left(\mathbf{f} \otimes \left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)^H\right)\left(\mathbf{1} \otimes \mathrm{vec}(\mathbf{R}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H)\right)\right) \\ &= \sum_{q \neq a} 2\mathbb{R}\mathrm{e}\left(\mathbf{f} \otimes \left(\left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a}\right)^H \mathrm{vec}(\mathbf{R}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H)\right)\right).\end{aligned} \tag{5.113}$$

This expression is identical to the $\mathbf{J}^H\breve{\mathbf{r}}$ component of the general phase-only solver with the notable exception of the Kronecker product. That being said, all the simplifications and

shortcuts we applied in the general phase-only case also apply here and we can write down the following final expression:

$$[\mathbf{J}_2^H \mathbf{J}_1^H \check{\mathbf{r}}]_a = \sum_{q \neq a} 2\,\mathbb{I}\mathrm{m}(\mathbf{f} \otimes \mathrm{diag}(\mathbf{G}_a^H \odot \mathbf{D}_{aq} \mathbf{G}_q \mathbf{M}_{aq}^H)). \tag{5.114}$$

We can expect any parametrisation of the phase to have a $\mathbf{J}^H \check{\mathbf{r}}$ of this form, provided it is linear with respect to its parameters and that the coefficients (e.g. t and ν) do not vary from one correlation to another.

The mixed product property is also of use in simplifying Equation 5.110:

$$\begin{aligned}[\mathbf{J}_2^H \tilde{\mathbf{H}} \mathbf{J}_2]_{aa} = &\sum_{q \neq a} \mathbf{f}\mathbf{f}^T \otimes \left(\left(\frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a} \right)^H \mathbb{W} \mathcal{R}_{\mathbf{M}_{aq} \mathbf{G}_q^H \mathbf{G}_q \mathbf{M}_{aq}^H} \frac{\partial \mathbf{g}_a}{\partial \boldsymbol{\phi}_a} \right) + \\ &\sum_{p \neq a} \mathbf{f}\mathbf{f}^T \otimes \left(\left(\frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a} \right)^H \mathbb{W} \mathcal{L}_{\mathbf{M}_{pa}^H \mathbf{G}_p^H \mathbf{G}_p \mathbf{M}_{pa}} \frac{\partial \mathbf{g}_a^\star}{\partial \boldsymbol{\phi}_a} \right).\end{aligned} \tag{5.115}$$

This is also very similar to the general phase-only case, and we can once again make use of the property described in Equation 5.77 as well as the structure of the derivative terms to write:

$$[\mathbf{J}_2^H \tilde{\mathbf{H}} \mathbf{J}_2]_{aa} = 2 \sum_{q \neq a} \mathbf{f}\mathbf{f}^T \otimes (\mathbf{I} \odot (\mathbf{M}_{aq} \mathbf{M}_{aq}^H)). \tag{5.116}$$

We are finally in position to write down a simple update rule for the phase-slope solver. First, however, we return to a point made at the start of the derivation - the omission of solution intervals. As they are a necessary component of this solver, we will reintroduce the subscript s in our final update rule. This yields:

$$\boldsymbol{\Delta\gamma}_a = \left(\sum_{q \neq a, s} \mathbf{f}_s \mathbf{f}_s^T \otimes (\mathbf{I} \odot (\mathbf{M}_{aqs} \mathbf{M}_{aqs}^H)) \right)^{-1} \sum_{q \neq a, s} \mathbb{I}\mathrm{m}(\mathbf{f}_s \otimes \mathrm{diag}(\mathbf{G}_a^H \odot \mathbf{D}_{aqs} \mathbf{G}_q \mathbf{M}_{aqs}^H)). \tag{5.117}$$

5.2.5 Solving for pointing errors

We have already established (see Chapter 3) that the antenna beams in conjunction with the per-antenna pointing errors introduce non-trivial direction-dependent effects into observations. This motivates our derivation of a pointing error solver.

The origin of the pointing errors is a mechanical mispointing which should affect all the feeds of a single antenna equally. Consequently, we make use of the scalar (single polarisation) version of our expressions noting that it should be possible to constrain the pointing error using only Stokes I. Note that we neglect the effects of the so-called beam squint (polarisation dependent pointing offsets) as they can be incorporated in the predict step assuming we have sufficiently accurate beam models.

Another unique aspect of this derivation is the fact that it requires, by necessity, direction-dependent information. This is due to the fact that the beam affects each source differently and we need to model that behaviour in order to constrain a solution. However, the pointing errors themselves are not direction-dependent. This is an excellent example of parametrisation

reducing the degrees of freedom of a calibration problem, as we ultimately solve for two offsets per antenna, as opposed to a unique direction-dependent gain per source. Our derivations for arbitrary parametrisations did not take this type of behaviour into account. However, it simply introduces a summation over directions into our expressions.

The minimisation problem can be expressed as follows:

$$\min_{\{\boldsymbol{\psi}_p\}} \sum_{pq} ||r_{pq}||_F, \qquad \text{where} \qquad r_{pq} = d_{pq} - \sum_{d=1}^{N_\text{D}} v_{d,pq}, \tag{5.118}$$

and,

$$v_{d,pq} = e_{d,p}(\boldsymbol{\psi}_p) m_{d,pq} \bar{e}_{d,q}(\boldsymbol{\psi}_q). \tag{5.119}$$

We have made a subtle change to the notation by replacing $g_{d,p}$ with $e_{d,p}$. This is to emphasise the fact that the $e_{d,p}$ terms represent the effect of antenna p's beam in direction d. The beam effects are parametrised by their pointing errors, $\boldsymbol{\psi}_p$, where each pointing error vector contains two offsets - one in the l direction and one in the m direction.

We define the following length N_A nested vector containing the per-antenna pointing error vectors:

$$\vec{\boldsymbol{\psi}} = \left[\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_{N_\text{A}}\right]^T = \left[\Delta l_1, \Delta m_1, \ldots, \Delta l_{N_\text{A}}, \Delta m_{N_\text{A}}\right]^T. \tag{5.120}$$

In order to proceed, we need the scalar equivalents of Equations 5.92 and 5.97. Obtaining these expressions is as simple as replacing any 2×2 term with its scalar equivalent. Additionally, we will include the summation over directions which we mentioned above. This summation is the result of having several directions sharing the same parameters. These changes yield:

$$\begin{aligned}[\mathbf{J}_2^H \tilde{\mathbf{H}} \mathbf{J}_2]_{aa} = &\sum_{q \neq a} \sum_{d=1}^{N_\text{D}} |m_{d,aq}|^2 |\bar{e}_{d,q}|^2 \left(\frac{\partial e_{d,a}}{\partial \boldsymbol{\psi}_a}\right)^H \left(\frac{\partial e_{d,a}}{\partial \boldsymbol{\psi}_a}\right) \; + \\ &\sum_{p \neq a} \sum_{d=1}^{N_\text{D}} |\bar{m}_{d,pa}|^2 |\bar{e}_{d,p}|^2 \left(\frac{\partial \bar{e}_{d,a}}{\partial \boldsymbol{\psi}_a}\right)^H \left(\frac{\partial \bar{e}_{d,a}}{\partial \boldsymbol{\psi}_a}\right)\end{aligned}, \tag{5.121}$$

and

$$[\mathbf{J}^H \check{\mathbf{r}}]_a = \sum_{q \neq a} \sum_{d=1}^{N_\text{D}} 2\,\mathbb{R}\text{e}\left(\left(\frac{\partial e_{d,a}}{\partial \boldsymbol{\psi}_a}\right)^H e_{d,q} \bar{m}_{d,aq} r_{aq}\right). \tag{5.122}$$

The derivative terms in the above expressions are scalar-by-vector derivatives and are given by:

$$\left(\frac{\partial e_{d,a}}{\partial \boldsymbol{\psi}_a}\right) = \left[\frac{\partial e_{d,a}}{\partial \Delta l_a} \quad \frac{\partial e_{d,a}}{\partial \Delta m_a}\right], \tag{5.123}$$

$$\left(\frac{\partial \bar{e}_{d,a}}{\partial \boldsymbol{\psi}_a}\right) = \left[\frac{\partial \bar{e}_{d,a}}{\partial \Delta l_a} \quad \frac{\partial \bar{e}_{d,a}}{\partial \Delta m_a}\right]. \tag{5.124}$$

These derivative terms are 1×2 vectors (or 2×1 under the conjugate transpose). Thus, the diagonal entries of the Hessian will be 2×2 matrices. It is also clear that the two components

of diagonal entries are conjugates of each other and we are free to simplify Equation 5.121, obtaining:

$$[\mathbf{J}_2^H \tilde{\mathbf{H}} \mathbf{J}_2]_{aa} = \sum_{q \neq a} \sum_{d=1}^{N_\mathrm{D}} 2\,\mathbb{R}\mathrm{e}\left(|m_{d,aq}|^2 |\bar{e}_{d,q}|^2 \left(\frac{\partial e_{d,a}}{\partial \boldsymbol{\psi}_a} \right)^H \left(\frac{\partial e_{d,a}}{\partial \boldsymbol{\psi}_a} \right) \right). \tag{5.125}$$

It is at this juncture that we have to note a lurking difficulty with this derivation - we do not usually have analytic expressions for the beams. Consequently, we cannot further simplify the derivative terms and usually have to rely on numerical differentiation of simulated/measured beam cubes.

Regardless, we can can still combine our expressions to write the following per-antenna pointing error update rule, noting that as in the general direction-dependent case, we are forced to use the residuals, r:

$$\boldsymbol{\Delta\psi}_a = \left(\sum_{q \neq a} \sum_{d=1}^{N_\mathrm{D}} 2\,\mathbb{R}\mathrm{e}\left(|m_{d,aq}|^2 |\bar{e}_{d,q}|^2 \left(\frac{\partial e_{d,a}}{\partial \boldsymbol{\psi}_a} \right)^H \left(\frac{\partial e_{d,a}}{\partial \boldsymbol{\psi}_a} \right) \right) \right)^{-1} \sum_{q \neq a} \sum_{d=1}^{N_\mathrm{D}} 2\,\mathbb{R}\mathrm{e}\left(\left(\frac{\partial e_{d,a}}{\partial \boldsymbol{\psi}_a} \right)^H e_{d,q} \bar{m}_{d,aq} r_{aq} \right). \tag{5.126}$$

Finally we note that this update rule is untested due to the difficulties involved in efficiently computing the beam derivatives. This is largely a computational and implementational limitation - there is every reason to expect that the update rule will work as intended.

Chapter 6

Implementation

Having derived several different solvers, it is pertinent to discuss CubiCal[1], the open source software package which implements them. CubiCal's documentation is available online[2] and both the code and the documentation should be considered digital appendices integral to this work. Additional information regarding the code and installation is provided in Appendix A.

6.1 Language and optimisation

CubiCal is implemented in Python[3], a high-level programming language popular in the astronomy community (some examples appear in Chapter 3). This popularity stems from its relative simplicity as a dynamically-typed language and its capacity for rapid prototyping. It is also highly extensible, and there are a wealth of Python modules dedicated to scientific computing, including NumPy[4] (Oliphant 2006), SciPy[5] (Jones et al. 2001–) and AstroPy[6] (The Astropy Collaboration et al. 2018). NumPy is used extensively within CubiCal.

Python supports many programming paradigms, the most relevant of which (in the context of CubiCal) is object-oriented programming (OOP). The core concept behind OOP is the "object" - a programmatic entity which can contain both data (referred to as attributes) and functions (referred to as methods). Typically, a program which makes use of the OOP paradigm will consist of several such objects which interact. We will return to this idea in a subsequent section.

It is also important to understand the limitations of Python. Unlike C or C++, Python code is not compiled. Instead, it is interpreted at runtime. This, coupled with type-checking (necessary for a dynamically-typed language), comes with a not insubstantial performance cost. Additionally, a fundamental limitation of CPython - the standard Python interpreter - is the presence of the Global Interpreter Lock (GIL). The GIL prevents pure Python code from being interpreted by more than a single thread. Unfortunately, this severely inhibits Python's ability to exploit multi-threading or leverage multi-core central processing unit (CPU) architectures.

[1] https://github.com/ratt-ru/CubiCal
[2] https://cubical.readthedocs.io
[3] https://www.python.org
[4] http://www.numpy.org
[5] https://www.scipy.org
[6] http://www.astropy.org

Python, however, does make it easy to incorporate code written in compiled languages such as C and C++. Extensions written in other languages are not bound by the GIL, and this remedies some of the aforementioned problems.

CubiCal makes use of Cython[7] (Behnel et al. 2011) for its extensions. Cython is both an optimising static compiler for Python and a superset of the Python language. Code written in either Python or Cython can be "cythonised" - turned into compilable C/C++ code. This approach maintains the readability of Python/Cython code but can offer C-like performance. Getting true C-like performance does require the addition of static type declarations, but that is usually a small price to pay for the increase in performance. Additionally, multi-threading is well supported in Cython. The components of CubiCal which make use of Cython were those found to be the slowest and most computationally intensive.

Whilst CubiCal does support multi-threading, we also took an alternative approach using multi-processing. Multi-processing allows several (independent) CPython interpreters to run simultaneously, each of which has its own GIL. This is an excellent strategy when the problem in question can be broken into several independent pieces which require similar processing, as each process can be assigned a single piece of the problem. Additionally, this approach means that we can include more GIL-limited pure Python without worrying about its performance implications.

Multi-processing has its own set of drawbacks, most of which are associated with inter-process communication (IPC) and concurrent memory accesses. Fortunately, the solvers we have derived can operate on subsets of the data completely independently and there is little to no need for IPC. However, we do have to worry about concurrent memory accesses as each process updates only a fraction of the total data. Additionally, multi-processing can lead to a large number of extraneous memory copies. This is a consequence of the fork system calls which are used to spawn the processes. We circumvent this problem by using shared memory. Shared memory is memory which can be read by and written to by several processes simultaneously.

6.2 Structure

CubiCal is, at its core, a very simple application. Whilst there is a fairly large amount of additional code and features (some of which have been implemented by a small but growing user-base), CubiCal only has three main components: a data handler, a solver and one of several gain machines.

6.2.1 Data handler

CubiCal's data handler is a Python object, the main function of which is to interface with the measurement set (the standard format for radio interferometric visibility data). We chose to make the data handler an object as it fits well within the OOP paradigm - the visibilities may be the data in the strictest sense, but there is also a host of metadata (such as times and channel frequencies) which we need to keep track of. Consequently, we store both the data

[7] http://cython.org

(or at the very least its location) and the relevant metadata as attributes on a data handler object. Of course, we will need to manipulate the data and so we also associate the data handler with several methods which do just that. Most importantly, this includes methods for reading from and writing to the measurement set.

In truth there is a second object which plays a part in the data handling component of CubiCal: the tile object. Due to the size of the data and our desire to exploit parallelism we rarely want to (or, indeed, have the ability to) read all the data from disk simultaneously. The tile object very literally tiles the data - a concept borrowed from MeqTrees - so that we need only deal with a single tile (usually consisting of a specific range of times and channels) at a time. However, a tile is still a large data structure. Internally, a tile consists of smaller time and frequency chunks and it is over these chunks that we parallelise when using multi-processing.

This two-tiered strategy - tiling and chunking - allows us to limit CubiCal's memory footprint with relative ease. Additionally, it provides us with another opportunity to accelerate the code by interleaving processing (calibration) with disk reads (I/O). Essentially, it is possible to process one tile whilst reading the next from disk. This allows us to effectively hide/offset the time taken by our I/O operations.

The tile object also includes functionality for generating predicted visibilities on a per-tile basis. This is accomplished using Montblanc[8] (Perkins et al. 2015) - a fast Python package which implements optimised CPU and GPU routines for evaluating the RIME. In order to support this functionality, CubiCal can read sky models. Note that this predict step, like the disk I/O described above, can be interleaved with compute i.e. we can also predict the contents of the next tile while processing the current one. This goes a long way to mitigating what has traditionally been one of the largest bottlenecks during calibration. This is particularly true in the context of direction-dependent calibration where it is necessary to predict N_{D} sets of model visibilities per tile.

6.2.2 Solver

The solver component of CubiCal is not an object. Instead, it is a fairly generic function which is used regardless of the gain which is being solved for. As such, its main function is to implement the basic iterative structure of the calibration algorithms as well as the ubiquitous bookkeeping - flagging and convergence testing.

There are several different modes of operation for the solver, but they are also generic - they could be applied to any type of gain. We are able to support this gain-agnostic solver by implementing the majority of the computation specific to a type of gain inside a so-called gain machine. The solver then only needs to manipulate these machines which have been designed with a unified interface.

[8] https://github.com/ska-sa/montblanc

6.2.3 Gain machines

A gain machine is another type of object implemented in CubiCal. Each gain machine consists of the attributes and methods necessary to perform a specific type of gain calibration. Of course, this means that there are several different types of gain machine. However, we make use of abstract base classes from the Python standard library and the OOP notion of inheritance (objects inheriting attributes and methods from other objects) to create a unified interface for all the gain machines.

Inheritance is an extremely useful property in this instance as it allows us to define several gain machine "layers". That is, we start from the most abstract implementation of a gain machine, one which does almost nothing bar define the attributes and methods which the solver function expects, and then inherit from it. The inheriting object can then modify the attributes and methods it inherited as well as add more. We can and do use inheritance several times, with each intermediary object becoming slightly more specialised until we arrive at a gain machine which implements the type of gain calibration we are interested in performing. The reason for this incremental specialisation is that several types of gain may require similar behaviour (similar ancestors/parents) up to some point of divergence.

A useful side-effect of both the gain-agnostic solver and the gain machines is that CubiCal is highly modular. Adding additional types of gain calibration is as simple as adding an additional gain machine. This provides CubiCal with an element of future-proofing, as we can incrementally add new and more sophisticated methods of gain calibration.

6.3 Data structures

An oft neglected topic when discussing implementation is the fundamental data structure which a piece of software employs. This choice of data structure can ease or hinder future development as well as have implications for performance.

The data structure we have adopted in CubiCal is based on Salvini and Wijnholds (2014) wherein they note that the StEFCal update rule can be written entirely in terms of products between rows and columns of so-called correlation matrices. The same is true of CubiCal's update rules (although we will express it differently) and the correlation matrices are the data structure which CubiCal employs. However, we will extend the definition to the 2×2 (4 correlation) case.

A correlation matrix is a matrix the entries of which include every possible baseline and its conjugate. That is, for any complex quantity which varies per baseline, the entries of its correlation matrix (denoted by $\hat{(\cdot)}$) are given by:

$$[\hat{\mathbf{A}}]_{ab} = \mathbf{A}_{pq} \qquad \text{where} \qquad p = a, q = b, \tag{6.1}$$

and each entry is a 2×2 matrix. The subscripts a and b index the rows and columns of the matrix as usual. We usually discard the autocorrelations during calibration, and consequently we set the diagonal entries ($a = b$) of this matrix to zero.

It is easier to understand these correlation matrices and their applicability to the problem with an example. Let us consider the single Jones term update rule which is equivalent to polarised StEFCal:

$$\mathbf{G}_{a,k+1} = \left(\sum_{q\neq a} \mathbf{D}_{aq}\mathbf{G}_q\mathbf{M}_{aq}^H\right)\left(\sum_{q\neq a} \mathbf{M}_{aq}\mathbf{G}_q^H\mathbf{G}_q\mathbf{M}_{aq}^H\right)^{-1}. \tag{6.2}$$

Our aim is to express this update rule in terms of correlation matrices. We will consider the simplest case of a three antenna interferometer although our conclusions are valid for any number of antenna. The per-baseline quantities in the above update rule are the model $\mathbf{M}$ and the data $\mathbf{D}$. Their correlation matrices are given by:

$$\hat{\mathbf{D}} = \begin{bmatrix} 0 & \mathbf{D}_{12} & \mathbf{D}_{13} \\ \mathbf{D}_{21} & 0 & \mathbf{D}_{23} \\ \mathbf{D}_{31} & \mathbf{D}_{32} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{D}_{12} & \mathbf{D}_{13} \\ \mathbf{D}_{12}^H & 0 & \mathbf{D}_{23} \\ \mathbf{D}_{13}^H & \mathbf{D}_{23}^H & 0 \end{bmatrix}, \tag{6.3}$$

$$\hat{\mathbf{M}} = \begin{bmatrix} 0 & \mathbf{M}_{12} & \mathbf{M}_{13} \\ \mathbf{M}_{21} & 0 & \mathbf{M}_{23} \\ \mathbf{M}_{31} & \mathbf{M}_{32} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{M}_{12} & \mathbf{M}_{13} \\ \mathbf{M}_{12}^H & 0 & \mathbf{M}_{23} \\ \mathbf{M}_{13}^H & \mathbf{M}_{23}^H & 0 \end{bmatrix}. \tag{6.4}$$

The gains themselves are antenna based, and we will reuse the nested vector notation of the previous section, noting that for three antennas:

$$\vec{\mathbf{g}} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \\ \mathbf{G}_3 \end{bmatrix}. \tag{6.5}$$

In order to proceed we need to define a new operator: the aligned product. Given two length N nested vectors, $\vec{\mathbf{b}}$ and $\vec{\mathbf{c}}$ - the entries of which are 2×2 matrices - the aligned product is defined as:

$$\vec{\mathbf{b}} \diamond \vec{\mathbf{c}} = \begin{bmatrix} \mathbf{B}_1\mathbf{C}_1 \\ \vdots \\ \mathbf{B}_N\mathbf{C}_N \end{bmatrix}. \tag{6.6}$$

The inputs to the aligned product need only be the same length - the operator does not care about their orientation. Note that the rows and columns of the correlation matrices can be treated as nested vectors and we are free to apply the aligned product to them.

Returning to the update rule, we can rewrite it as follows:

$$\mathbf{G}_{a,k+1} = \left(\sum_{2\times 2} \hat{\mathbf{D}}_{a,:} \diamond \vec{\mathbf{g}}^H \diamond \hat{\mathbf{M}}_{:,a}\right)\left(\sum_{2\times 2} \hat{\mathbf{M}}_{a,:} \diamond \vec{\mathbf{g}} \diamond \vec{\mathbf{g}}^H \diamond \hat{\mathbf{M}}_{:,a}\right)^{-1}, \tag{6.7}$$

where the colon in the subscript indicates the selection of all entries within the a'th row or column. We draw attention to the fact that summation over baselines is now just a summation over the 2×2 blocks of the resulting nested vectors. This shows that our update rule can easily be computed as a product of nested vectors.

Practically, we do not update the gain of each antenna independently as it is not as efficient as doing larger matrix operations. However, we can simultaneously apply Equation 6.7 to all antennas using broadcasting. Broadcasting (which is well implemented in NumPy), refers to performing element-wise operations between matrices with different but compatible dimensions. We will not attempt to derive a notation for broadcasting here, as it would be needlessly complicated. The point we are trying to make is that we can exploit the correlation matrices in conjunction with broadcasting provided by NumPy to implement our update rules both efficiently and simply.

It is also worth pointing out that the correlation matrix structure we have presented does not include times, frequencies or directions. All of these come into play as we extend our update rules to include solution intervals and direction-dependence. Fortunately, this is in fact another point in favour of the correlation matrices; the additional dimensions can be added as leading dimensions and many of the summations in the update rules will just require collapsing the additional axes.

Using the correlation matrices has worked well thus far, although we must acknowledge the fact that the data structure is not as compact as it could be. This is due to the fact that we store the conjugate entries and the zero-valued cross-correlations. However, conjugation is itself quite an expensive operation as it involves a memory copy and we do gain some efficiency at the cost of a larger memory footprint.

Chapter 7

Simulations

Having discussed the details of both the mathematics and implementation of CubiCal, it only remains to demonstrate its speed and efficacy. This chapter will present the results of applying CubiCal to simulated data.

We chose to perform our simulations for the field surrounding 3C147, a compact, bright source (quasar). This field is often used as a high dynamic-range imaging challenge (Mitra et al. 2015; Perley 2013; Smirnov 2011c) due to the high brightness of 3C147 itself - very accurate calibration and imaging are required to produce noise-limited results. Additionally, the real data to which CubiCal will be applied in a subsequent chapter is also from this field. Consequently, we will be able to make comparisons between the simulated and observed cases. We go as far as using an identical measurement set and (almost identical, see below) sky models to ensure that our results are consistent.

The measurement set in question is the product of a C-configuration, Karl G. Jansky Very Large Array (VLA) observation. The target field was observed for a little under six hours with an integration time of five seconds. For the sake of consistent comparison, we restrict our simulations to the portion of the band which is relatively RFI-free in the real data. This sub-band was divided into 64 channels with channel widths of 4MHz, ranging between 1.2665GHz and 1.5815GHz. An average of 26 antennas were present (unflagged) for the duration of the observation.

We made use of two different sky models during the simulations: one containing sources and their intrinsic (true) flux values and the other containing apparent (beam-attenuated) flux values. This was necessary in order to demonstrate the effect of the primary beam and will be explained in greater detail during the individual experiments. Note that the sky model of the 3C147 field we used was the same as that employed by Smirnov (2011c) and was originally obtained from VLBI measurements conducted by A. G. de Bruyn.

One change was made to the sky models used for the simulated case - 3C147 itself was removed. This was done to ensure that we could produce visually interesting results (3C147 otherwise completely dominates the images) as well as to ensure that CubiCal works correctly in lower signal-to-noise regimes (the remaining sources are far fainter). This is also a more challenging calibration problem, as in the real data 3C147 both dominates and tightly constrains the gain solutions.

All the simulated data was produced using MeqTrees (Noordam and Smirnov 2010) and the relevant sky model. Following the removal of 3C147, the sky model contained 55 of the

brighter remaining sources. The errors included in the simulated data will be discussed in detail for each experiment.

7.1 A note on data products

The majority of our results will be presented as images which are suitable for qualitative comparison. We made use of WSClean (Offringa et al. 2014) to perform the imaging. However, we make several images for each experiment and some of the terminology which we use to describe our results deserves further explanation and contextualisation. To do so, let us consider the following, hopefully familiar, expression:

$$\mathbf{R}_{pq} = \mathbf{D}_{pq} - \mathbf{G}_p \left(\sum_{d=1}^{N_\mathrm{D}} \mathbf{E}_{d,p} \mathbf{M}_{d,pq} \mathbf{E}_{d,q}^H \right) \mathbf{G}_q^H, \tag{7.1}$$

where $\mathbf{R}_{pq}$ is the residual visibility associated on baseline pq, $\mathbf{D}_{pq}$ is the visibility observed by baseline pq and $\mathbf{M}_{d,pq}$ is the model visibility predicted from the sky model in direction d. $\mathbf{G}$ and $\mathbf{E}$ are direction-independent and direction-dependent gains (or products thereof) respectively.

First and foremost, we will present images of the inputs to CubiCal. In the above expression, the inputs correspond to the observed data $\mathbf{D}$ and predicted/model data $\mathbf{M}$. Images of these quantities in the upcoming experiments are labelled as the model and (observed) data respectively. Bear in mind that Equation 7.1 describes a single baseline and that our images will be made using the visibilities of every baseline for all times and frequencies.

The first output image we consider is the corrected data, $\mathbf{D}'_{pq}$. This is obtained by multiplying the observed data by the inverse of the gain terms:

$$\mathbf{D}'_{pq} = \mathbf{G}_p^{-1} \mathbf{D}_{pq} \mathbf{G}_q^{-H}. \tag{7.2}$$

The absence of the direction-dependent gain terms in this expression is deliberate and underlines a limitation of direction-dependent gain calibration: it is impossible to correct a visibility with respect to more than one direction simultaneously. Practically, the direction-dependent gains can only be applied as a correction during deconvolution using faceting (e.g. DDFacet, Tasse et al. (2018)) or A-projection (Bhatnagar et al. 2013). CubiCal is not yet integrated with an imager which supports either such functionality. Thus, we omit images of the corrected data when using direction-dependent gains.

The second output we consider is the (direction-independent) corrected residual, $\mathbf{R}'_{pq}$. This is obtained from the following:

$$\mathbf{R}'_{pq} = \mathbf{G}_p^{-1} \mathbf{R}_{pq} \mathbf{G}_q^{-H} = \mathbf{G}_p^{-1} \mathbf{D}_{pq} \mathbf{G}_q^{-H} - \left(\sum_{d=1}^{N_\mathrm{D}} \mathbf{E}_{d,p} \mathbf{M}_{d,pq} \mathbf{E}_{d,q}^H \right). \tag{7.3}$$

As in the case of the corrected data, there is some difficulty when direction-dependent gains are present as they cannot be applied to the residual or observed data terms with ease. However, we can still image the direction-independent corrected residual as described

by Equation 7.3. This is usually the best diagnostic of the quality of direction-dependent gain calibration - the closer to noise the direction-independent corrected residual looks, the better the direction-dependent calibration.

Finally, before presenting the individual experiments, we must stress that we do not deconvolve any of the images which we produce. This is done to ensure that we do not conflate deconvolution effects with calibration effects. Consequently, all the images to follow will be so-called "dirty" images.

7.2 Experiment 1

The aim of our first experiment was to verify that CubiCal works in the simplest case. We simulated observed visibilities using the apparent sky model, corrupted by a slowly-varying (periodic in amplitude and phase) complex gain term and noise based on the SEFD (system equivalent flux density) of the VLA at L-band.

We applied CubiCal to this simulated data, feeding in the apparent sky model for the model prediction step. Recall that model prediction in CubiCal is handled by Montblanc (Perkins et al. 2015). We solved for a single, direction-independent gain term ($\mathbf{G}$) with a solution interval of 20s and 16MHz. The use of a solution interval was necessary due to the relatively low SNR of the data. The topic of optimal solution intervals is an area of active research (Mbou Sob et al., in prep.).

The four images which appear in Figure 7.1 are the inputs and results of this experiment. The top left image corresponds to model visibilities predicted from the apparent sky model, uncorrupted by either the gain or noise. The top right image is of the simulated observed data. Note how the gain errors have made the individual sources almost indistinguishable. The corrected data image appears on the bottom left. There is clearly a dramatic improvement following correction by the gains, as can been seen by comparing this image with that of the model. Note that the presence of noise in the data prevents them from being identical. Finally, the fourth image corresponds to the corrected residual. This image is noise-like which means that CubiCal successfully corrected for the gain errors we introduced.

The results of all the subsequent experiments will share the layout of Figure 7.1 unless otherwise noted. This does mean that some images will be repeated. We permitted this redundancy to make comparison at a glance possible.

7.3 Experiment 2

The second experiment is similar to the first. However, we now consider the effect of the primary beam on the observed visibilities. We already know from Chapter 1 that antennas are neither perfect nor equally sensitive in all directions. This introduces direction-dependent phase and amplitude errors into the observed data. In addition, the direction-dependent effects vary with frequency as the primary beam itself varies as a function of frequency. Finally, parallactic angle rotation introduces a dependence on time: over the course of an

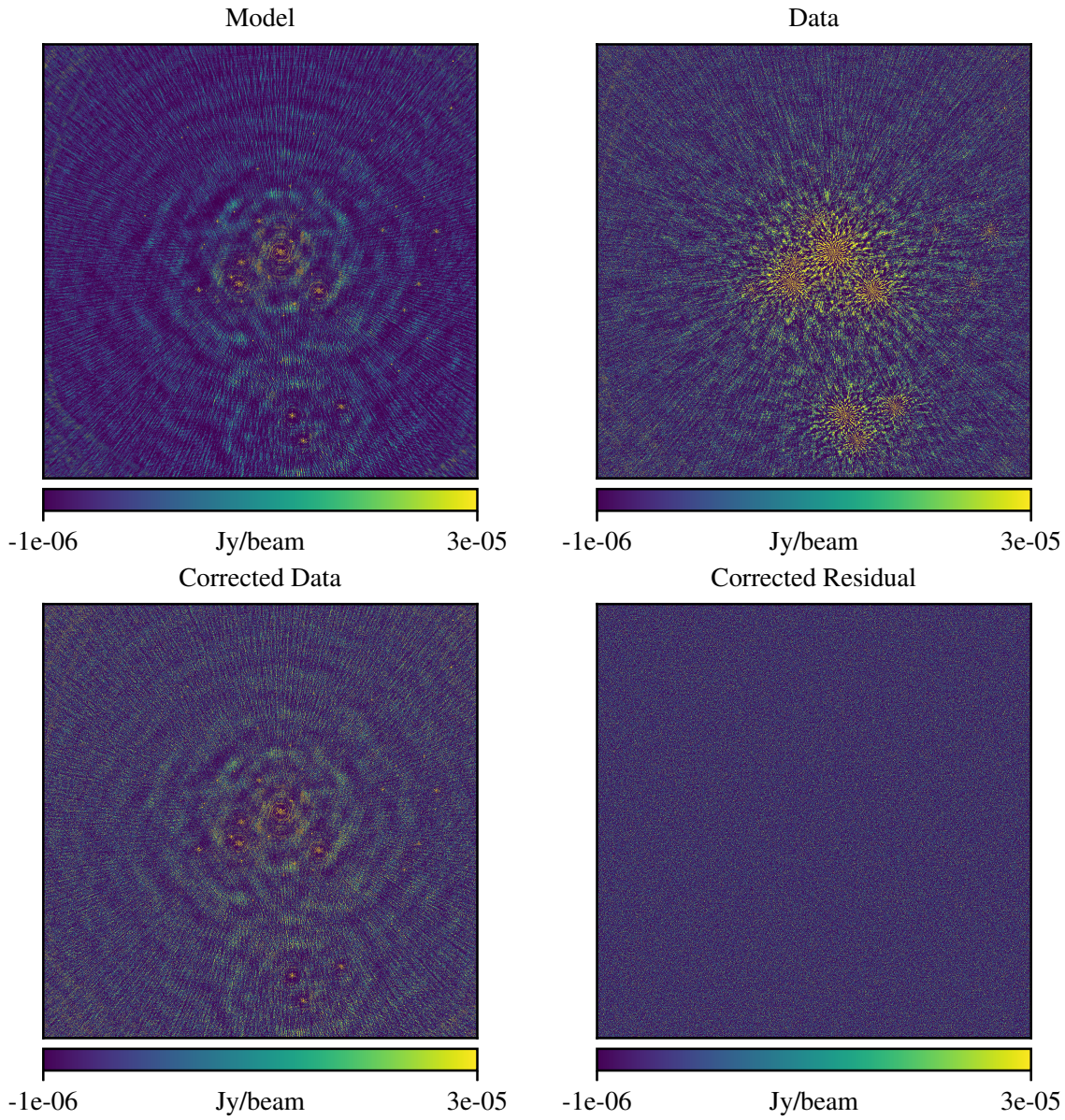

FIGURE 7.1: Images of the simulation inputs and outputs for the first experiment. *Top left:* Image of the model visibilities obtained from the apparent flux sky model. *Top right:* Image of the observed data which includes the contribution of a DI gain and noise. *Bottom left:* Image of the corrected data after DI gain calibration. *Bottom right:* Image of the residuals after DI gain calibration.

observation, sources move relative to the primary beam. These effects are more thoroughly discussed in Smirnov (2011b) and Mitra et al. (2015).

Naturally, given that the primary beam is ubiquitous, it is important to be able to correct for the errors it introduces. To demonstrate CubiCal's capacity in this regard, we simulate visibilities using the intrinsic sky model and a beam cube (cube of images containing beam patterns at various frequencies). This beam information was obtained from ray tracing simulations of the VLA beam (Brisken 2003). As before, we also introduce a complex gain term and noise.

We apply CubiCal to this case using the apparent sky model, i.e. the predict step does not take the beam into account explicitly. We continue to solve for a single, direction-independent gain term ($\mathbf{G}$) over the same solution interval.

The inputs and results of the second experiment appear in Figure 7.2. The top left image corresponds to model visibilities predicted from the apparent sky model, uncorrupted by either the gain or noise. This image is identical to the model image produced in the first experiment. The top right image is of the simulated observed data. Whilst this is very similar to that of the first experiment, close inspection reveals differences introduced by the time and frequency variable beam gain. The image on the lower left is the corrected data, obtained following the correction of the data using the gains determined by CubiCal. The correction has only been partially successful, as evidenced by the spoke-like artefacts surrounding sources away from the field centre. The trio of sources towards the bottom edge of the image are particularly noticeable. This imperfect calibration is consistent with what we expect - we only solve for a single gain term per antenna. This approach cannot accurately model the direction-dependent behaviour of the beam. The fourth and final image is of the corrected residuals. The regions where the direction-independent term is inadequate are clearly visible.

7.4 Experiment 3

The third experiment is identical to the second with one exception: we now exploit CubiCal's Jones chains to solve for a direction-dependent gain ($\mathbf{E}$) in addition to the direction-independent gain term ($\mathbf{G}$). This replicates the MeqTrees-based approach taken by Smirnov (2011c) and Perley (2013). The sky model contains ten sources flagged for direction-dependent gain calibration clustered into seven unique directions. We solve for the direction-dependent gains using a solution interval of 80s and 64MHz. We use a larger interval to reduce the risk of over-fitting and to ensure we have sufficient SNR in each direction to constrain a gain solution. The solution interval of the direction-independent term is left unchanged.

The inputs and results of the third experiment appear in Figure 7.3. The images on the top row are identical to those of the second experiment. We present them again to make comparison easier. The image on the bottom row corresponds to the residuals after correction by the direction-independent gain. As previously mentioned, we cannot correct the residual in each direction as we cannot split the observed visibilities by direction. However, we can and do subtract the model multiplied by both the direction-dependent and direction-independent gains from the data before applying the direction-independent correction. There is a clear

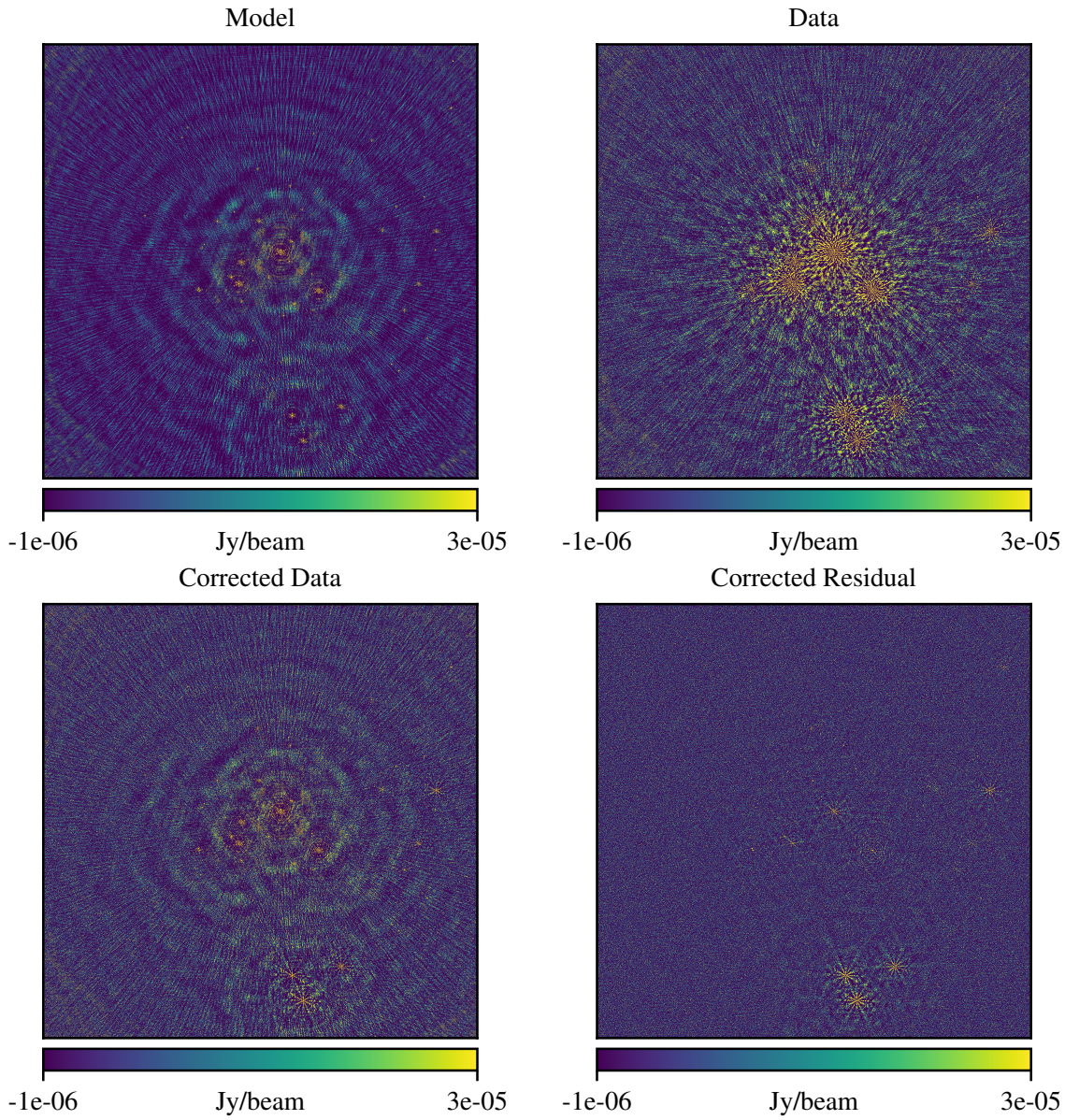

FIGURE 7.2: Images of the simulation inputs and outputs for the second experiment. *Top left:* Image of the model visibilities obtained from the apparent flux sky model. *Top right:* Image of the observed data which includes the contribution of a periodic DI gain, the primary beam and noise. *Bottom left:* Image of the data after DI gain calibration. *Bottom right:* Image of the residuals after DI gain calibration.

improvement in the residual map - the spoke-like artefacts have disappeared. However, there is still some obvious structure/flux in the residuals. This is due to the fact that the gains are, at best, piece-wise approximations of the true gain. This imperfect modelling results in slight inaccuracies in the gain solution which in turn leads to some flux being left in the residuals. Additionally, we only solve for direction-dependent gains towards a subset of the sources in the model. However, the beam affects all sources and consequently we do not expect perfectly noise-like residuals.

7.5 Experiment 4

The fourth experiment is once again very similar to the second. However, we now supply CubiCal (and consequently Montblanc) with both the intrinsic sky model and a primary beam model. We also return to solving for a single direction-independent gain per antenna. This replicates the MeqTrees-based approach taken by Mitra et al. (2015) and Makhathini (2018), providing an important consistency check.

Figure 7.4 contains the inputs and results of this experiment. The top left image corresponds to the visibilities predicted from the intrinsic sky model modified by the primary beam. In this case we have augmented the model with additional information in order to make calibration easier. By incorporating the beam in this experiment, we account for the direction-dependent component without needing to resort to direction-dependent gains (and the increase in degrees of freedom). The top right image is the same as in second and third experiments - the observed data has not changed. The bottom left image shows the corrected data following gain calibration. There is a dramatic improvement in the quality of the image but there are still spoke-like artefacts around some sources. However, this is precisely what we expect as those artefacts are also in the model as a result of the beam (see the first image in Figure 7.4). This, in conjunction with the fact that the final image is noise-like, means that the calibration was successful. Removing the effect of the beam (the spoke-like features) is as simple as adding the model image obtained from the intrinsic sky model (without the beam) to the residual image.

7.6 Experiment 5

This experiment differs from those preceding it in that we wish to assess whether or not a specialised solver, specifically the delay solver, works. We simulated observed visibilities in the same fashion as the first experiment. However, instead of adding a complex gain, we incorporated a slope in phase across frequency for each antenna (a delay). The delays were chosen to follow a normal distribution centred on 0 with a standard deviation of 0.5ns (1ns $= 1 \times 10^{-9}$s). We did not include a phase offset in this simulation as the effect of the slope is of greater interest.

Figure 7.5 contains the inputs and results of the fifth experiment. The top left image corresponds to the visibilities predicted from the apparent sky model. The top right image is of the simulated observed data. Note how the delays have had a smearing effect. This is

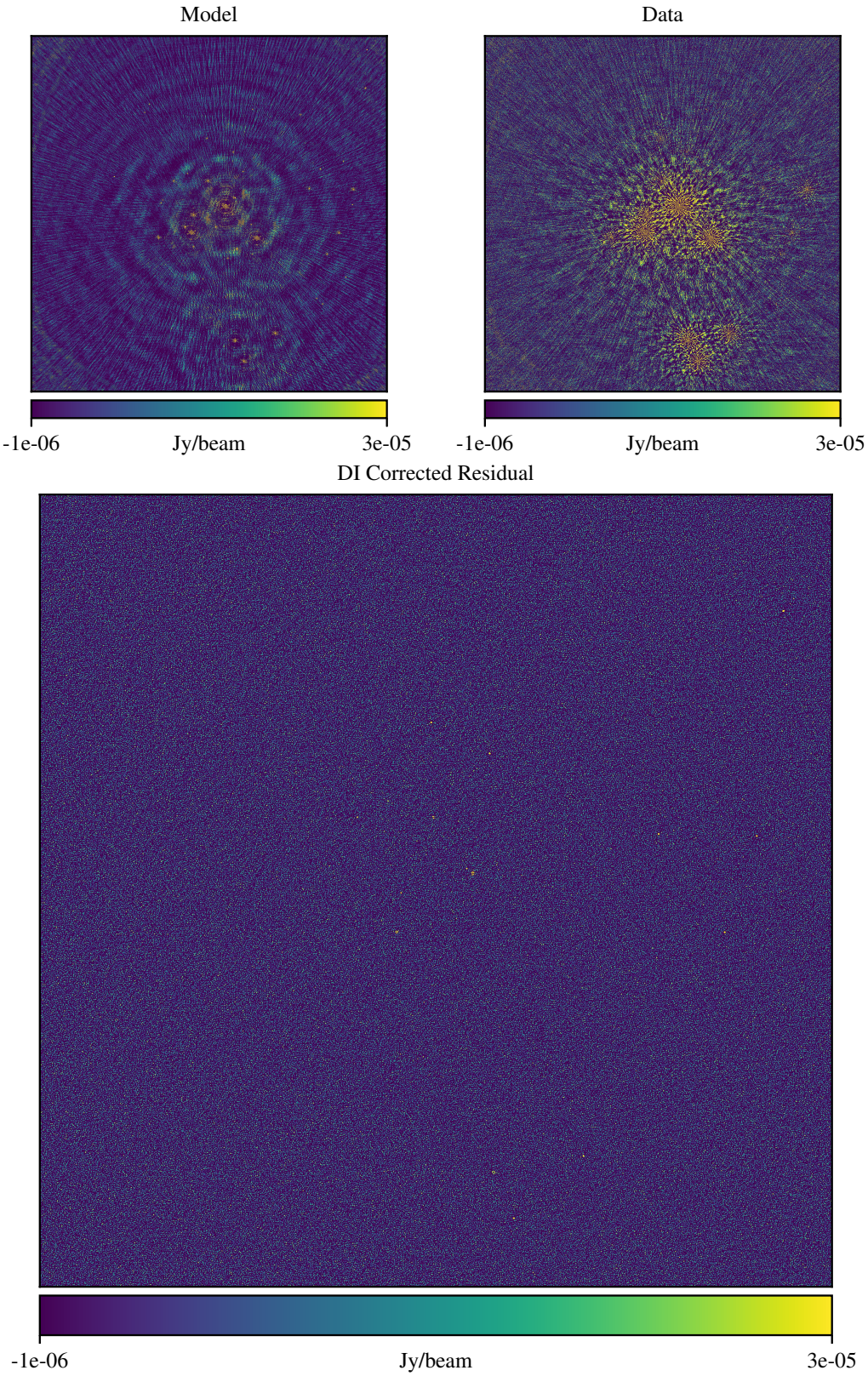

FIGURE 7.3: Images of the simulation inputs and outputs for the third experiment. *Top left:* Image of the model visibilities obtained from the apparent flux sky model. *Top right:* Image of the observed data which includes the contribution of a periodic DI gain, the primary beam and noise. *Bottom:* Image of the residuals after DI and DD gain calibration.

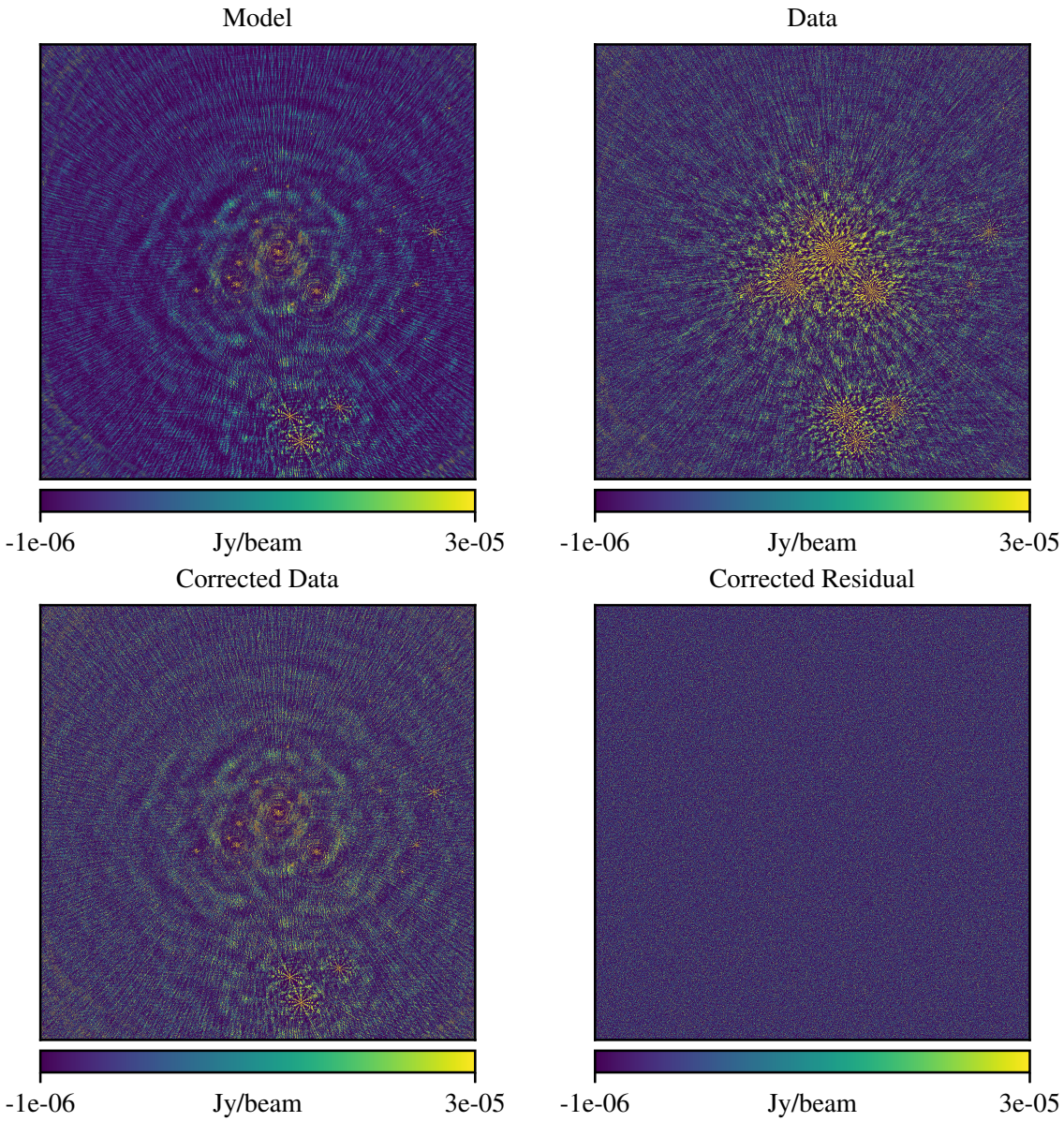

FIGURE 7.4: Images of the simulation inputs and outputs for the fourth experiment. *Top left:* Image of the model visibilities obtained from the intrinsic flux sky model modified by the primary beam. *Top right:* Image of the observed data which includes the contribution of a periodic DI gain, the primary beam and noise. *Bottom left:* Image of the data after DI gain calibration. *Bottom right:* Image of the residuals after DI gain calibration.

due to the fact that the delays corrupt the phase and phase encodes position information in Fourier space. The bottom left image is the data after delay calibration using CubiCal. Once again, it is very similar to the model image with the exception of the noise. The final residual image on the bottom right is completely noise-like. This confirms that the delay solver works and suggests that our method of constructing specialised solvers functions correctly.

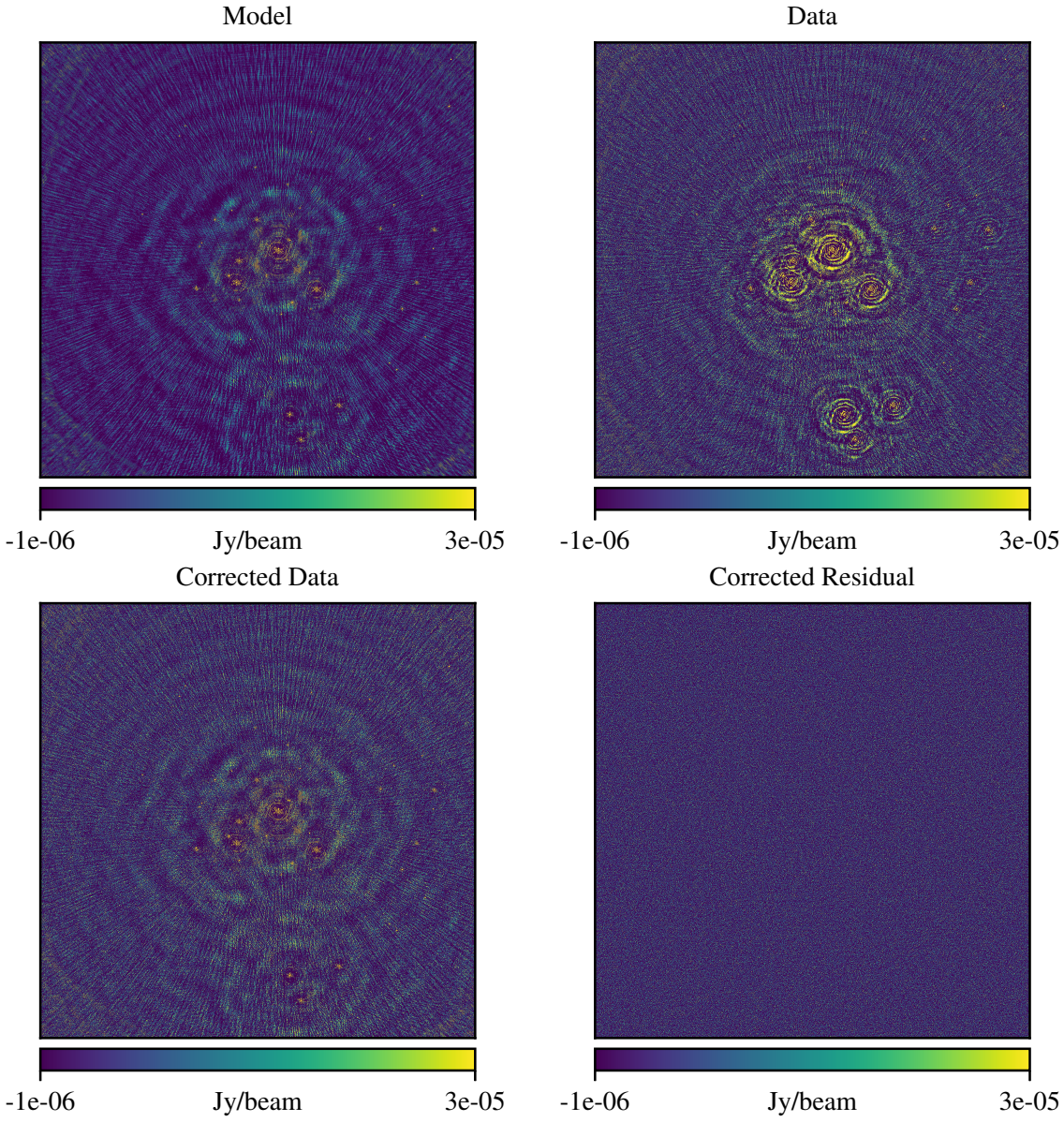

FIGURE 7.5: Images of the simulation inputs and outputs for the fifth experiment. *Top left:* Image of the model visibilities obtained from the apparent flux sky model. *Top right:* Image of the observed data which includes the contribution of delay and noise. *Bottom left:* Image of the data after delay calibration. *Bottom right:* Image of the residuals after delay calibration.

Chapter 8

Application to Real Data

Having shown that CubiCal works for simulated data, it is necessary to show that it can cope with real data and the various difficulties it brings. We will continue to use the measurement set described in the preceding chapter but we will now consider the actual observed visibilities. As before, we will perform several different experiments to demonstrate CubiCal's efficacy in a variety of cases.

Before doing so, we will present some of our general observations about the experiments. The first such observation is very simply that 3C147 is present in the data, and thus it is necessary to include it in our model. However, as was intimated in Chapter 7, it is so bright that prior to deconvolution it (and its PSF sidelobes) completely swamp all other emission in the field. Figure 8.1 is an image of the observed data prior to self-calibration and in the absence of deconvolution. Only very close inspection reveals even a hint of some of the other emission in the field.

As we mentioned previously, any gain solution we obtain will be both dominated and tightly constrained by 3C147. This behaviour is not surprising when we inspect an image of the intrinsic sky model (see Figure 8.2). It is almost impossible to discern a difference between this image of the model visibilities and the image of the data (Figure 8.1), highlighting the extent to which we expect 3C147 to impact our gain solutions. We have neglected to present an image of the apparent sky as it is just as dominated by 3C147 and provides no additional insight into the problem.

The dominance of 3C147 has another repercussion - it makes images of the corrected data (when we are in a position to obtain them) largely inadequate as a means of determining CubiCal's performance. Even with perfect calibration (almost impossible due to imperfect models, low-level RFI and sundry other small errors) we do not expect other emission to be visible in the field without deconvolution. We still wish to avoid conflating deconvolution and calibration errors and, as a result, we concentrate only on the dirty images of the corrected residuals. 3C147, as part of the model, is subtracted in these images, revealing the bulk of the other sources visible in the field. As in the previous chapter, this approach should make calibration errors fairly clear.

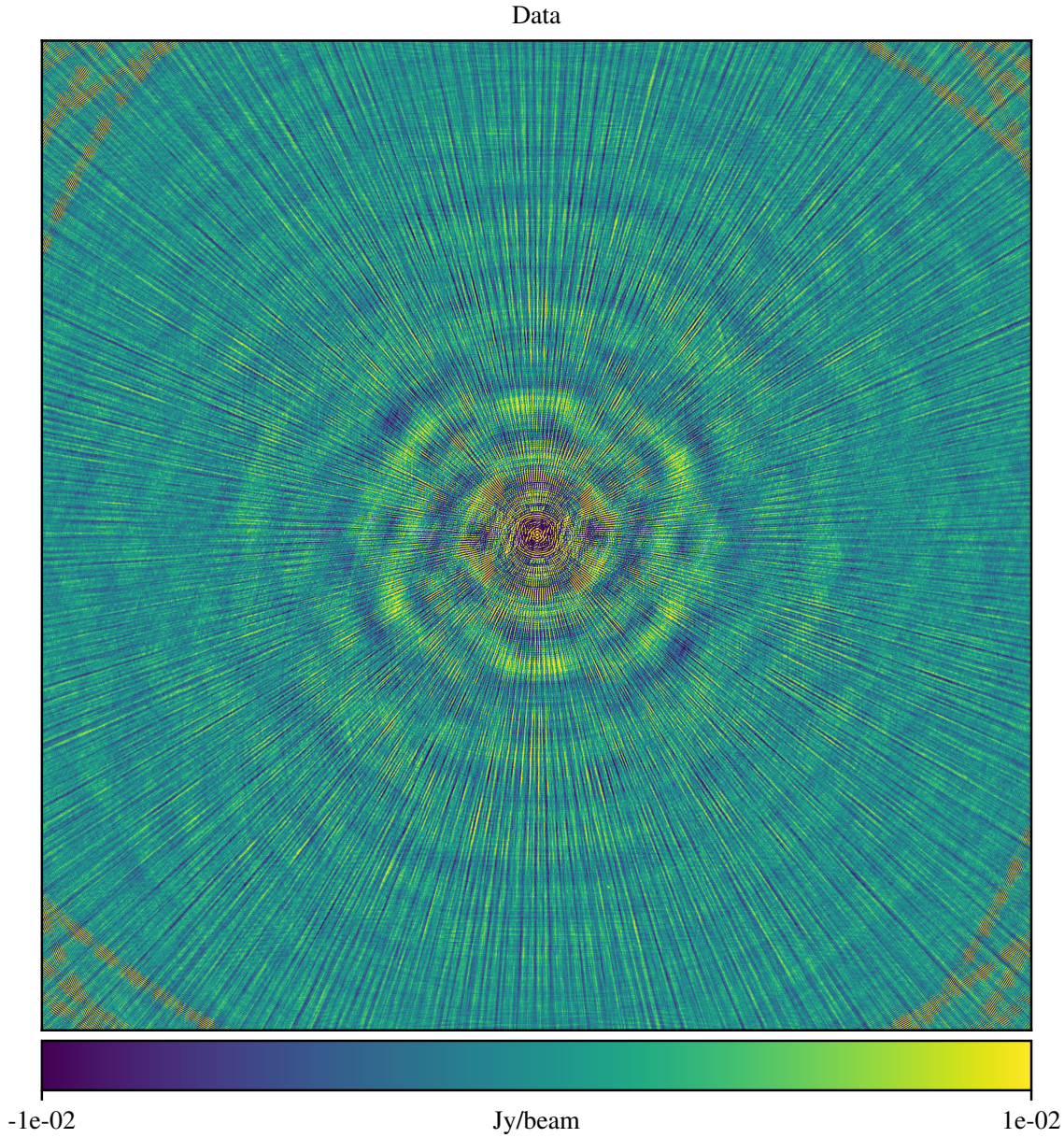

FIGURE 8.1: Image of the observed data. The central source, 3C147, dominates the image to such an extent that all other emission in the field is hidden beneath its PSF sidelobes.

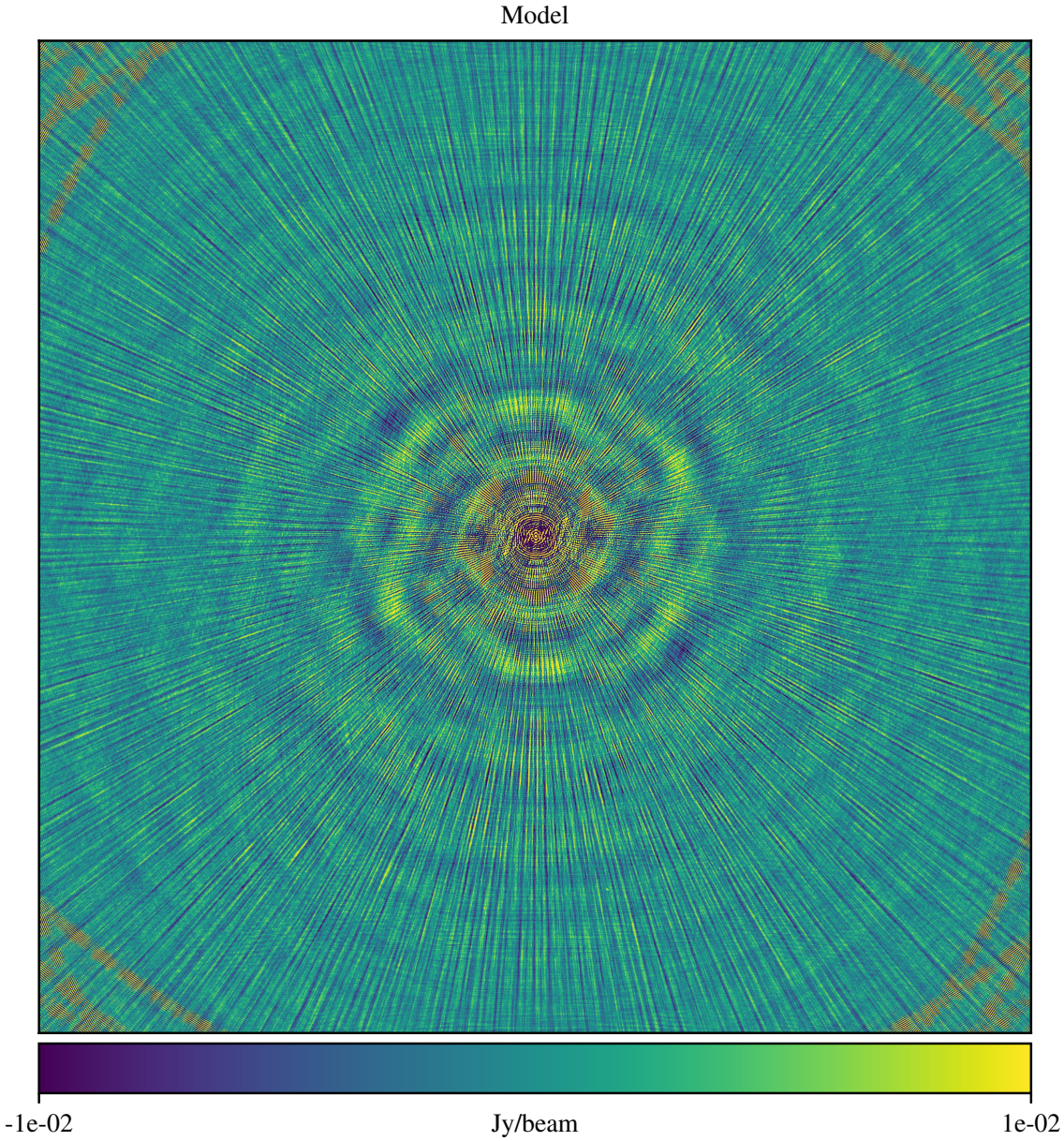

FIGURE 8.2: Image of the model visibilities associated with the intrinsic SM. It is difficult to discern any difference between the image of the data and the image of the model due, once again, to the dominance of 3C147 itself.

8.1 Experiment 1

The first experiment we performed using the real data has the most in common with the second experiment of Chapter 7. We use an apparent sky model and solve for a single direction-independent gain (**G**), although we reduce the solution interval to a single integration (5s) and channel (4MHz). This reduction in solution interval is both motivated and justified by the presence of 3C147. The concerns we had regarding signal-to-noise are absent in the real data and we want to calibrate as accurately as possible to ensure that 3C147 can be properly subtracted from the data.

Figure 8.3 is an image of the corrected residuals after calibration. The familiar spoke-like artefacts are once again apparent, particularly around sources away from the field centre. As we saw in the simulations, these artefacts are, at least partially, the result of the primary beam. This will be pursued in subsequent experiments.

Another striking feature of Figure 8.3 when compared to the contents of Figure 7.2, is the large number of sources in the field which are not in the model. Naturally, these sources appear in the residual as they are not subtracted from the data. In practice, the residual map could be used to build up a deeper model for subsequent self-calibration cycles.

Even though we used the shortest possible solution interval, there is still a degree of over-subtraction at the position of 3C147 (the centre of the image). This can be attributed to slight inaccuracies in the model we used. 3C147 itself is slightly extended at the resolution of this observation and modelling it accurately using only point sources and Gaussians (the norm in current software, including Montblanc) is challenging.

8.2 Experiment 2

This experiment is a simple extension of the first, and parallels the third experiment of the preceding chapter. In addition to our direction-dependent (**G**) term, we also solve for a direction-dependent (**E**) term. Again, this replicates the MeqTrees-based approach taken by Smirnov (2011c) and Perley (2013). As in the simulated case, 10 sources, clustered into 7 unique directions, were tagged for direction-dependent gains. Our concerns regarding signal-to-noise and over-fitting are applicable in the direction dependent-case and we choose to solve the direction-dependent gains over an 80s and 64MHz solution interval. This is the same solution interval that was used during our experiments on simulated data.

The direction-independent corrected residuals appear in Figure 8.4. The majority of the obvious artefacts around the brighter sources have disappeared. In particular, the spoke-like artefacts caused by the beam seem to have been completely eliminated around the sources to which we applied direction-dependent gains. Close inspection of the image does, however, show that some of the sources tagged for direction-dependent solutions are not perfectly subtracted. As we mentioned in the third experiment in Chapter 7, this is largely due to the direction-dependent gains being a piece-wise approximation of the true gains. What remains in the residual is the signature of effects which we did not completely remove.

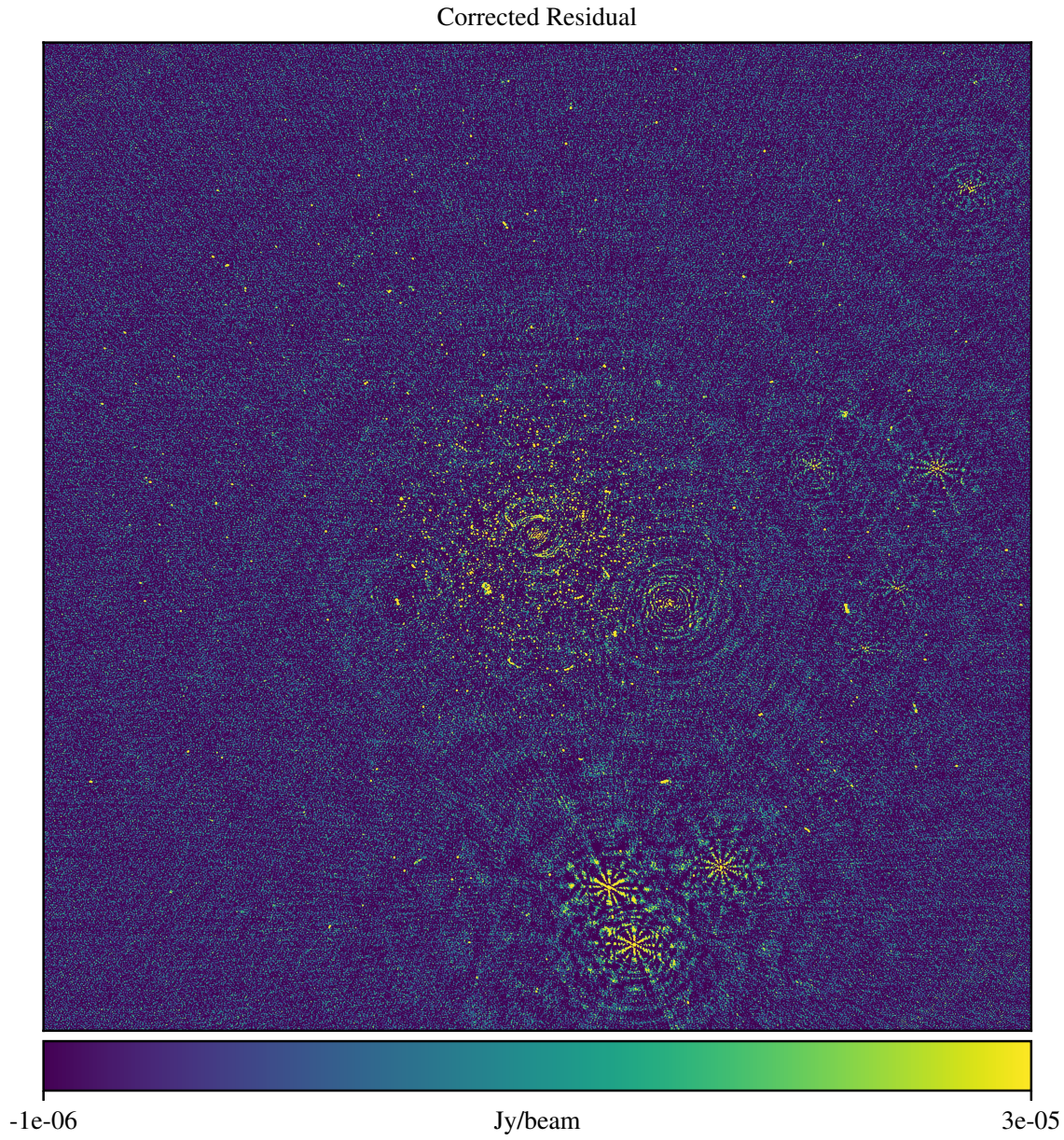

FIGURE 8.3: Image of the corrected residuals after DI calibration using an apparent SM (experiment 1). The spoke-like artefacts observed in the simulated case are clearly visible, particularly around sources away from the field centre. Note that, unlike the simulated case, there are far more sources in the field than are in the SM.

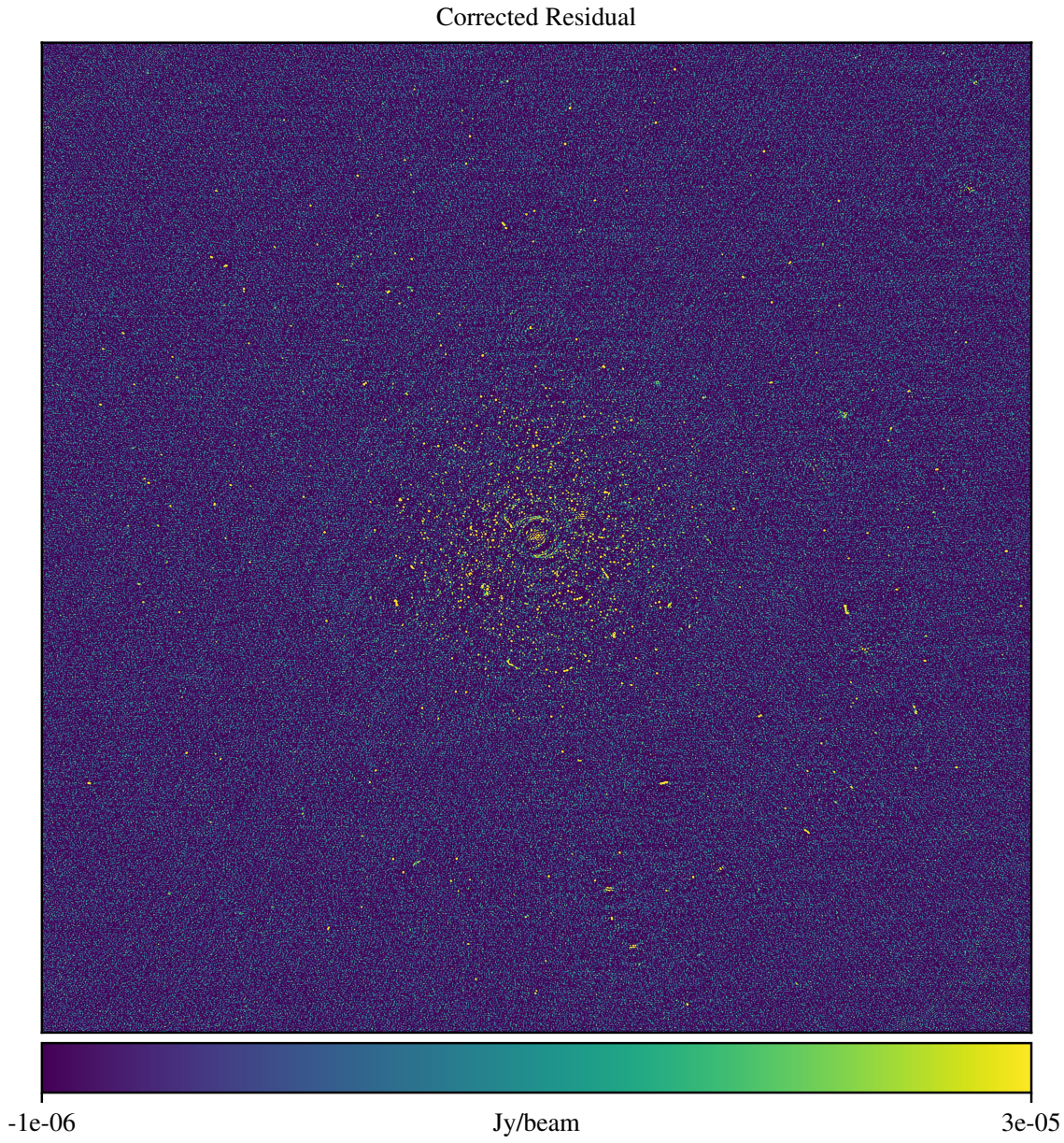

FIGURE 8.4: Image of the DI-corrected residuals after both DI and DD calibration using an apparent SM (experiment 2). The spoke-like artefacts have been dramatically reduced and there is an overall improvement in source subtraction. However, as the model is incomplete and the gains are solved over a solution interval, there is still some residual contribution from sources tagged for DD calibration.

8.3 Experiment 3

For the third experiment, we include our a priori knowledge of the beam from ray-tracing simulations (Brisken 2003). This replicates the MeqTrees-based approach taken by Mitra et al. (2015) and Makhathini (2018). In order to incorporate the beam, we use an intrinsic sky model and apply the beam during the prediction step of calibration. We then repeat the first experiment of this chapter.

The corrected residuals for this experiment appear in Figure 8.5. Comparison of this image with that of Figure 8.3 shows a reduction in artefacts, particularly around sources away from the field centre or, in other words, the sources most affected by the beam. However, this is a considerably less remarkable improvement than we saw in the simulated case (see Figure 7.4). This is most probably due to discrepancies between the simulated beam pattern and the true beam pattern. Additionally, there are also other sources of direction-dependent effects in the real data that were absent in the simulations.

8.4 Experiment 4

The failure of the approach taken in the previous experiment to completely eliminate the obvious artefacts in the image motivated this experiment, one which we did not perform on the simulated data. We continue to incorporate our a priori knowledge of the beam, but we also solve for a direction-dependent gain ($\mathbf{E}$) in precisely the same way as we did in the second experiment.

The direction-independent corrected residual produced by this experiment appears in Figure 8.6. As in Figure 8.4, the majority of the obvious artefacts are eliminated. In fact, the two images are very similar. This is encouraging, as it means that we can adequately handle the direction-dependent effects introduced by the beam without necessarily having a beam model. That being said, it is always preferable to include as much a priori knowledge as possible, if only to make disentangling the various sources of error simpler. The obvious advantage of applying the beam is that it can be applied to every source in the model (or field, if the model is complete). Solving for a direction-dependent gain toward every source in the field would be some combination of impractical and impossible due to a proliferation of degrees-of-freedom.

8.5 Experiment 5

This, the final experiment of the chapter, aims to demonstrate CubiCal's ability to solve for several direction-independent terms simultaneously. We omitted this in the simulated case for the sake of brevity. We will solve for a bandpass term ($\mathbf{B}$) and a generic complex gain term ($\mathbf{G}$). This is common practice in self-calibration. The bandpass is expected to be stable with time and consequently has high resolution in frequency and low resolution in time. The complex gain term has low resolution in frequency but high resolution in time. For the purposes of this experiment, the bandpass was solved per scan and the complex gain was

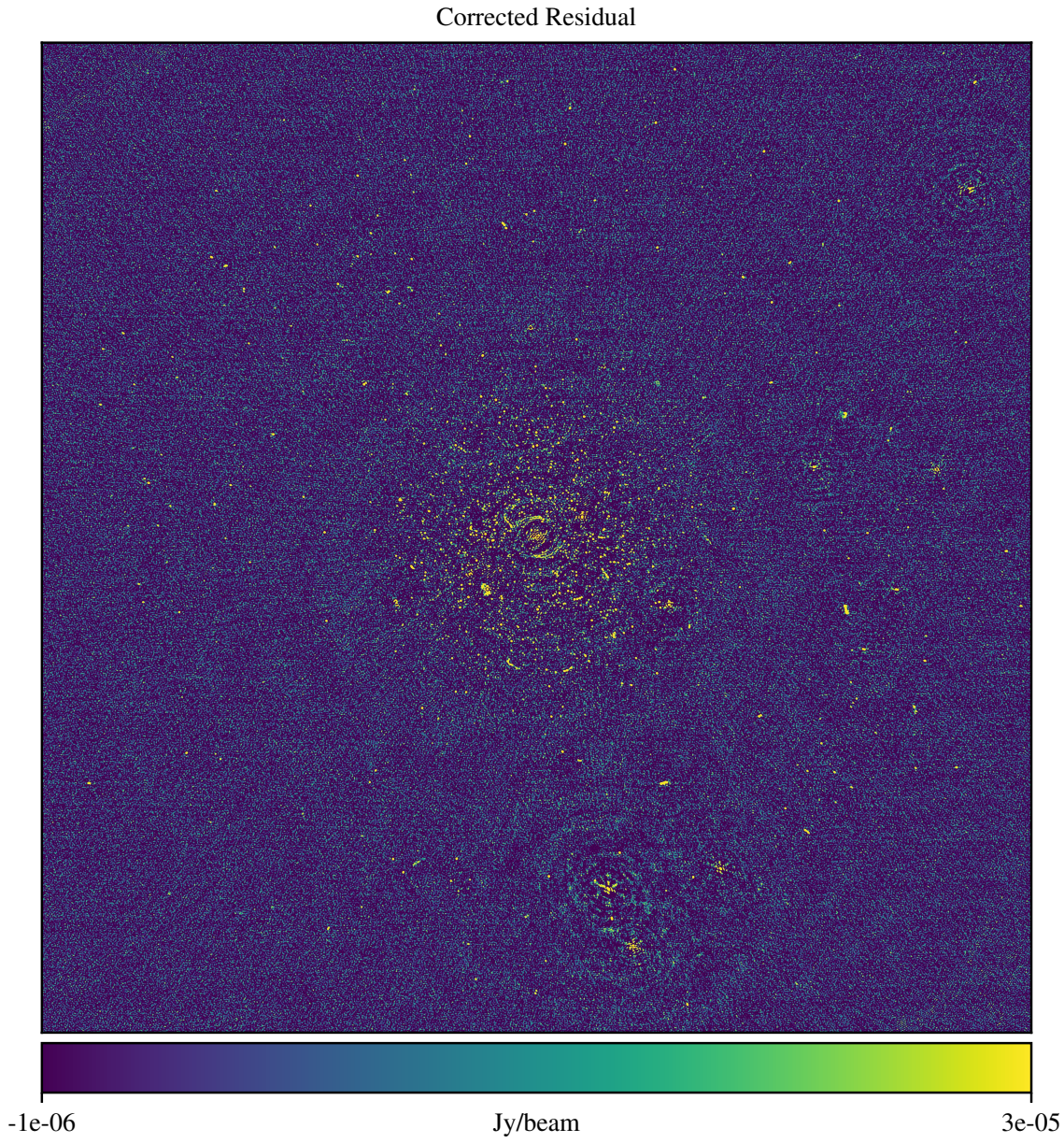

FIGURE 8.5: Image of the corrected residuals after DI calibration using an intrinsic SM and the beams (experiment 3). The spoke-like artefacts are not as pronounced, as they have been partially corrected by the beam. However, as the beams are neither perfectly accurate, nor the only source of DD errors in the observed data, there are still artefacts around sources away from the field centre.

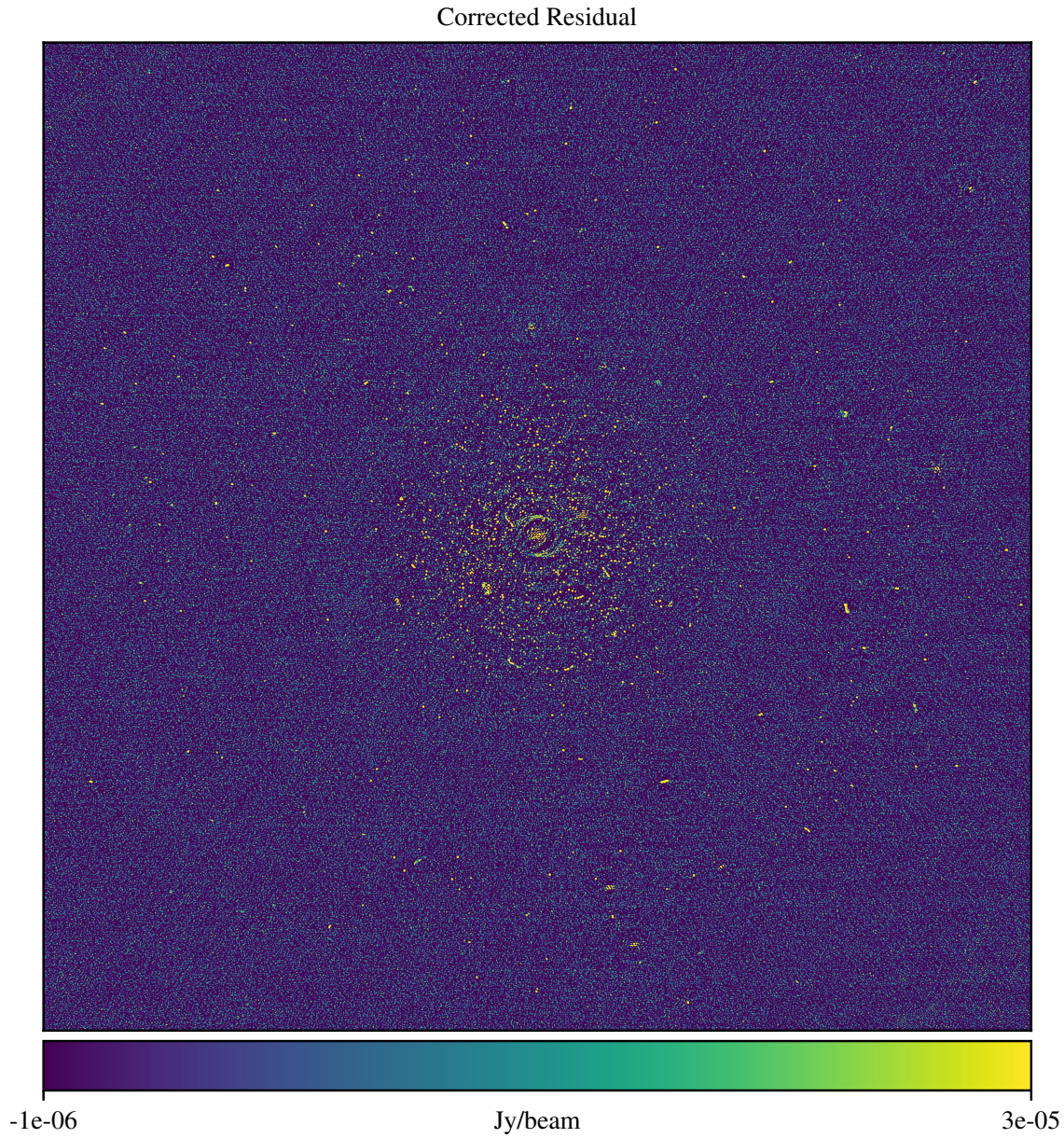

FIGURE 8.6: Image of the DI corrected residuals after both DI and DD calibration using an intrinsic SM and the beams (experiment 4). There is a clear improvement to the residual image. In fact, this residual image is slightly superior to that obtained using the apparent SM as, unlike DD gains, the beam can be applied to every source in the model.

solved over the entire bandwidth per integration. Note that we revert to using the apparent sky model.

The corrected residual for this experiment appears in Figure 8.7. Comparing this to Figure 8.3, we can see many of the same artefacts. Figure 8.7 is, however, slightly inferior in general, as evidenced by the slightly higher noise. This is not unexpected as we made use of far fewer degrees-of-freedom than in the first experiment.

Let us consider a single scan, consisting of N_t integrations and N_ν channels. In the first experiment, we solved for $\mathbf{G}$ without a solution interval, producing $N_t N_\nu$ gain solutions. In this experiment, $\mathbf{B}$ was solved per-channel over the entire scan, producing N_ν gain solutions. $\mathbf{G}$ for this experiment was solved per integration over the whole bandwidth, producing N_t gain solutions. Thus, the total number of gain solutions for this experiment was $N_t + N_\nu$, which is substantially smaller than the $N_t N_\nu$ solutions obtained in the first experiment.

It is quite impressive that it is possible to achieve similar results using what amounts to orders of magnitude fewer degrees-of-freedom. Additionally, this approach is far less likely to introduce calibration artefacts such as ghosts (Grobler et al. 2014, 2016; Wijnholds et al. 2016) in the presence of an incomplete sky model due to the large solution intervals. These solution intervals also ensure that we have sufficient signal-to-noise to constrain our solutions. The above arguments also explain why this approach to direction-independent gain calibration is so ubiquitous.

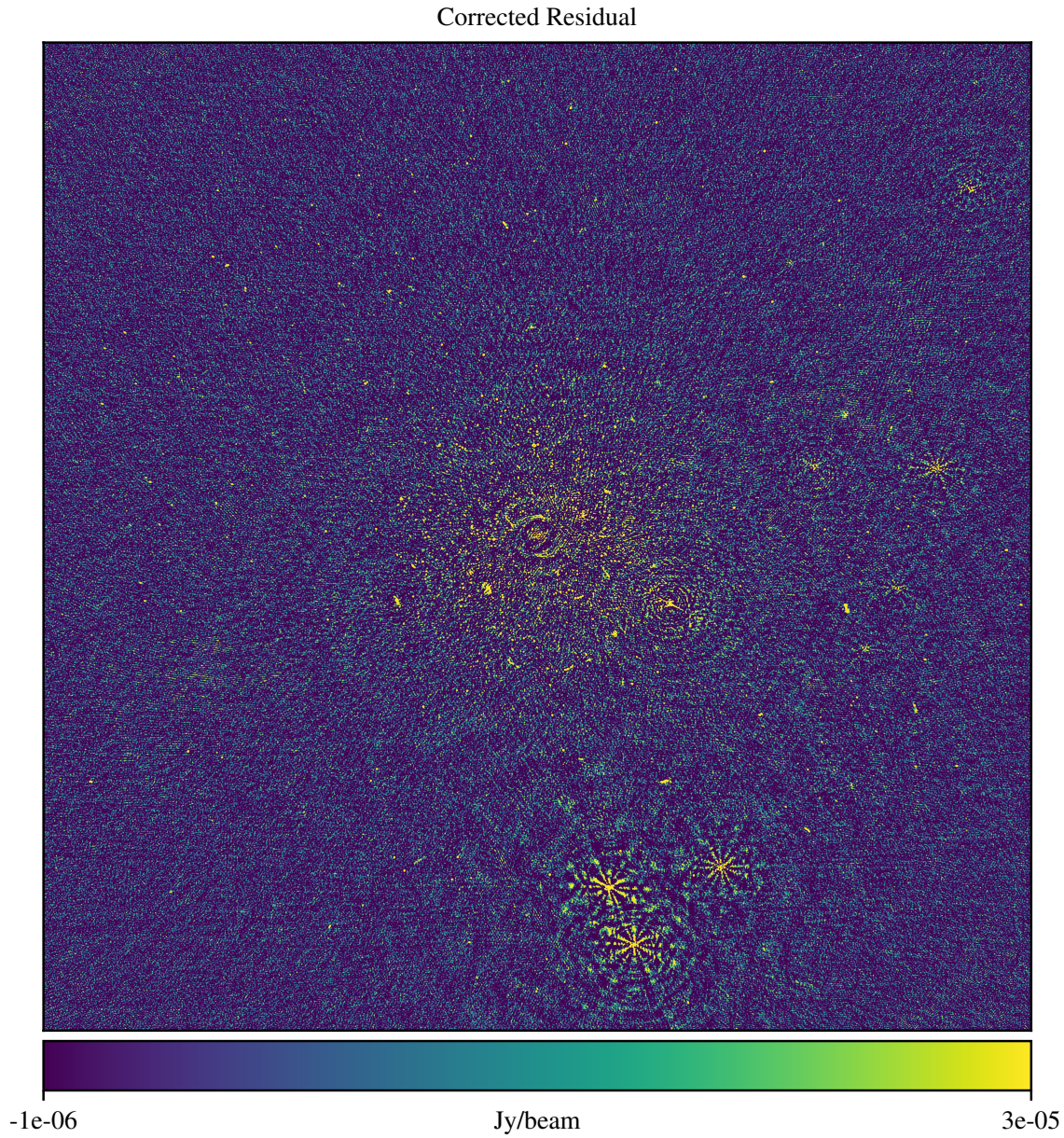

FIGURE 8.7: Image of the corrected residuals after DI calibration using an apparent SM (experiment 5). This differs from the first experiment as two DI terms were used - a complex gain term **G** and a bandpass term **B**. The residual in this case is slightly worse than that obtained in the first experiment, due to huge reduction in degrees-of-freedom.

Chapter 9

Performance

Finally, we need to substantiate our claims about CubiCal's speed. To that end, we will perform some relatively simple benchmarking tests. However, before we do so, it is necessary to describe the environment in which we performed these benchmarks, including both our test data and the hardware we used. Some of the results used in this chapter are adapted from Kenyon et al. (2018).

9.1 Hardware and Environment

All the benchmarking experiments we undertook were performed on a RATT (Radio Astronomy Techniques and Technologies) server housed at Rhodes University. The server was running two Intel® Xeon® E5-2695 v4[1] CPUs (18 cores, 36 threads per processor), the core frequencies of which vary between 2.10GHz and 3.30GHz as a result of Intel® Turbo Boost Technology. This dynamically adjusts core frequency based on power-consumption, temperature and load. Unfortunately, this technology does have some repercussions for profiling as there is no guarantee that the processor speed is the same between experiments. However, some of the profiling utilities we used also produced an average processor speed. Where possible, we used these values to normalise all timing values to a 3.10GHz core speed.

The system had 512GB of RAM (random-access memory) running at 2400MHz, which was more than sufficient for our tests. All data was stored on platter drives (relatively slow) - we did not perform experiments using SSDs (solid state drives) but would expect them to reduce disk I/O overheads. The server was running Ubuntu 16.04 LTS.

It is worth mentioning that the server in question was contested - we were not the sole users at the time of testing. That being said, every attempt was made to perform experiments during periods of low contention and each experiment was repeated several times to ensure that the results were approximately consistent. The advantage of testing in this environment is that our results are conservative estimates of CubiCal's performance - in practice, users should equal or exceed these values for sensible input parameters.

The data we used during these experiments was once again the VLA observation of the 3C147 field. The observed and model visibility columns of the measurement set each contain around 3.5×10^8 visibilities, corresponding to around 2.8GB of data per column on disk. As problem sizes go, this is not a large one. However, it is large enough to provide insight into

[1] https://ark.intel.com/products/91316/

CubiCal's speed and scaling whilst being small enough to carry out benchmarks in a realistic amount of time.

One additional point which applies to all the experiments is that we do not profile the prediction component of the solution. This is because it is handled by Montblanc (Perkins et al. 2015) and profiling a third-party module is not the purpose of this chapter. Practically, we exclude the prediction step by populating the model column of the measurement set prior to (as opposed to during) calibration. The timings obtained for MeqTrees also exclude the contribution of the predict step.

9.2 Experiment 1

It is, unfortunately, difficult to construct a fair comparison between the most common pieces of software implementing calibration. This is due to the fact that almost none of them implement the same algorithm, and the few that do do not have identical input parameters or parallelism. Consequently, we elected to use MeqTrees' (Noordam and Smirnov 2010) four correlation StefCal implementation as our comparison/reference, as it is essentially identical to using a single, direction-independent **G** term in CubiCal. However, unlike CubiCal, MeqTrees only makes use of parallelism during the model predict step, which we have excluded from these tests. Thus, we cannot make any definitive claims about MeqTrees' scaling. However, the MeqTrees implementation does provide an estimate of what would be considered typical speed for an existing tool in the absence of parallelism.

In addition to a comparison with MeqTrees, we want to investigate the impact of data time and frequency chunk dimensions on performance. CubiCal's parallelism is a result of multiprocessing (see Chapter 6) and we expect the dimensions of the data chunks submitted to each process to have a marked effect on performance. Note that we do not consider the case where there are more processes than available CPU cores/threads.

In order to investigate these effects, we calibrated the data several times keeping our input parameters the same with two exceptions: the number of processes used and the time and frequency dimensions of the data chunks submitted to those processes. This produces an average execution time as a function of chunk dimensions and number of processes. Figure 9.1 and Figure 9.2 are two different representations of the results.

The first figure, Figure 9.1, shows the actual time in seconds taken as a function of the number of processes used. Additionally, each coloured line corresponds to different chunk dimensions. The dotted line at the top of the figure corresponds to the time taken by MeqTrees to perform an almost identical calibration.

It is clear that CubiCal's parallelism works, as there is a rapid decrease in execution time as we increase the number of processes. However, there is clearly an underlying asymptotic limit which we approach at and beyond 16 processes. This is the disk I/O bound - we cannot calibrate faster than the time it takes to read all the data from disk plus any associated overhead. As a simplistic check of this, we did a bulk (all values loaded simultaneously) read of the measurement set columns used by CubiCal. This took around 15 seconds, which is close to the lowest execution times (around 25 seconds). The minor discrepancy is due to the

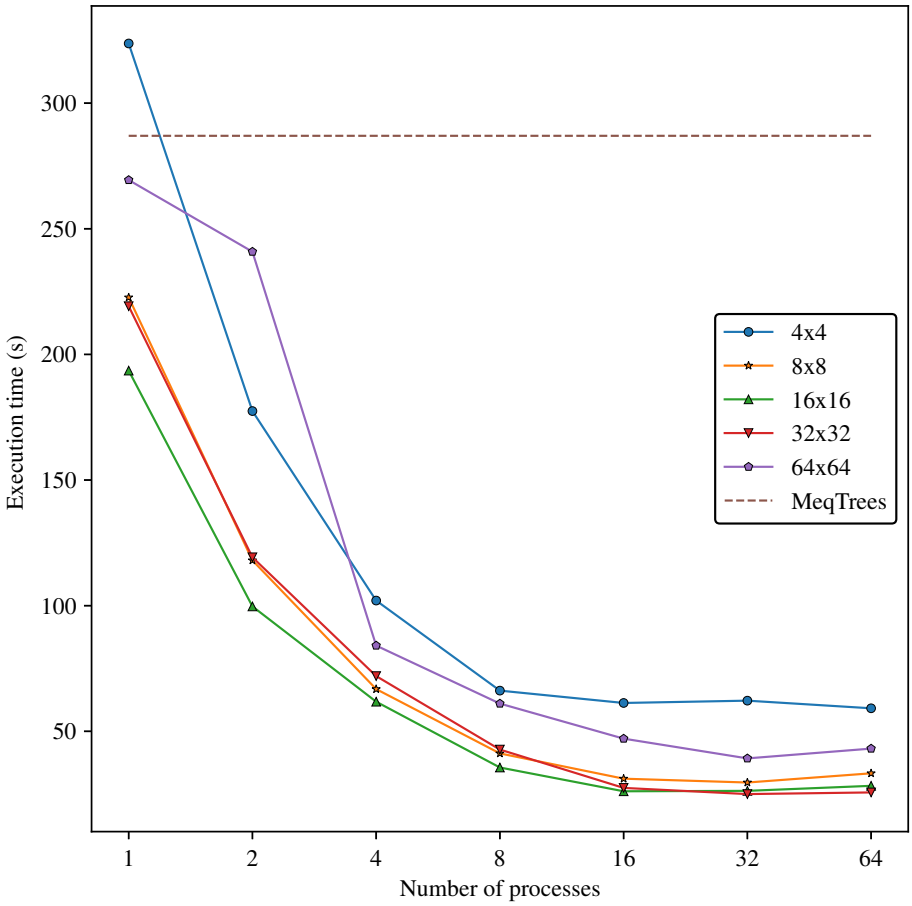

FIGURE 9.1: Plot of execution time against number of calibration processes. Each solid line corresponds to different chunk dimensions. The dashed line is the execution time of the reference single-core MeqTrees implementation. Figure adapted from Kenyon et al. (2018).

fact that in practice we do not do a bulk read - we read the data in tiles (see Chapter 6) - and there is some additional overhead due to pre- and post-processing.

The increase in CubiCal's performance is perhaps easier to interpret in Figure 9.2, which is CubiCal's speed-up for each chunk size and number of processes relative to the MeqTrees implementation. For optimal CubiCal input parameters, we achieve an order of magnitude improvement over the MeqTrees implementation. One minor fact is omitted from both of the presented figures: the number of processes used is one larger than appears on the x-axis. This extra process performs no calibration - instead, as discussed in Chapter 6, it handles all I/O.

Both figures show that CubiCal performs at its best when the time-frequency chunks have dimensions (16,16) and (32,32). There are a number of interacting effects which go some way to explaining why performance is optimal for these dimensions. The first of these is simply that, unlike the (4,4) case, at these chunk dimensions each process is doing sufficient work to minimise overheads incurred when chunking the data. It is also possible that we have superior cache behaviour at these chunk dimensions and we will pursue this in a subsequent experiment. Our suspicions regarding cache behaviour are somewhat borne out by the fact

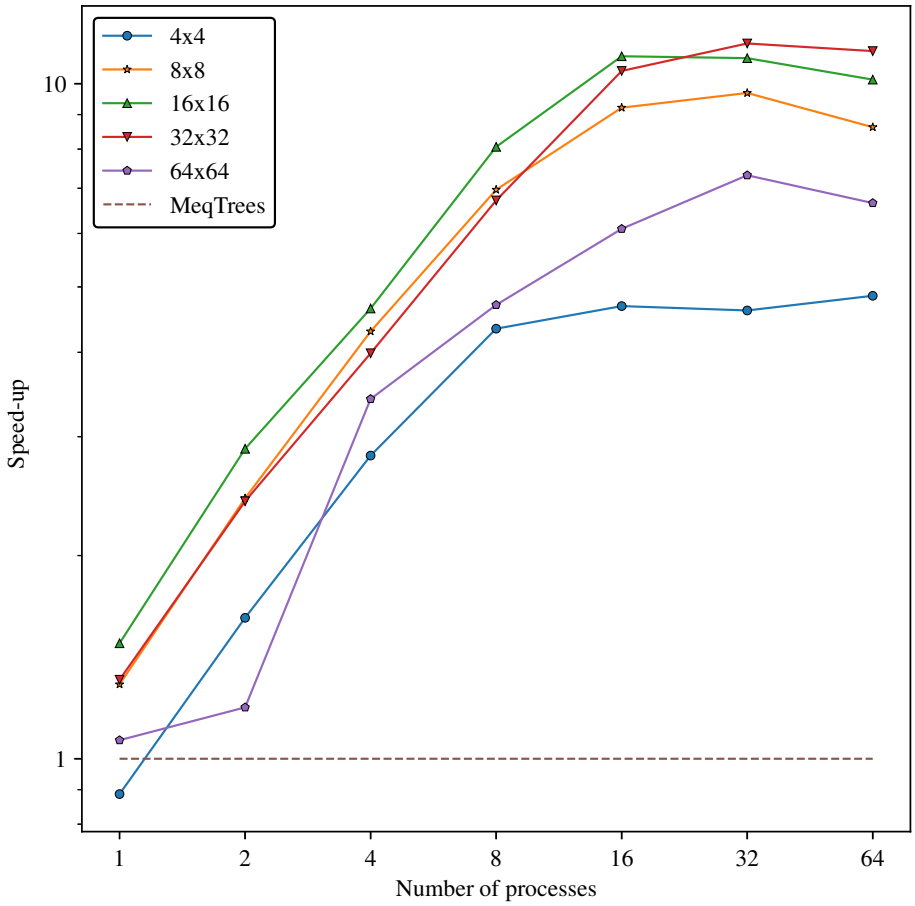

FIGURE 9.2: Plot of speed-up against number of calibration processes. Each solid line corresponds to different chunk dimensions. The dashed line is the speed of the reference single-core MeqTrees implementation. Figure adapted from Kenyon et al. (2018).

that for (64,64) chunk dimensions there is a sudden degradation in performance.

9.3 Experiment 2

The goal of our second experiment was to provide additional insight into CubiCal's cache behaviour as a function of chunk dimensions. To this end, we wanted to record the percentage of all cache references which resulted in a cache miss. Every cache miss causes the CPU to look for the referenced element in higher level caches (slower) or in RAM. Thus, we expect worse performance as cache misses increase, and we expect an increase in cache misses as our problem grows too large to fit in cache.

We made use of perf[2], a profiling tool available in Linux which gives us access to CPU performance counters. These performance counters track a variety of hardware events such as cache and branch misses. Branch misses occur when the CPU incorrectly predicts the next instruction to process. The branches themselves occur at any point where program flow can continue in a number of ways (a conditional statement, for example). As we were already

[2] https://perf.wiki.kernel.org/

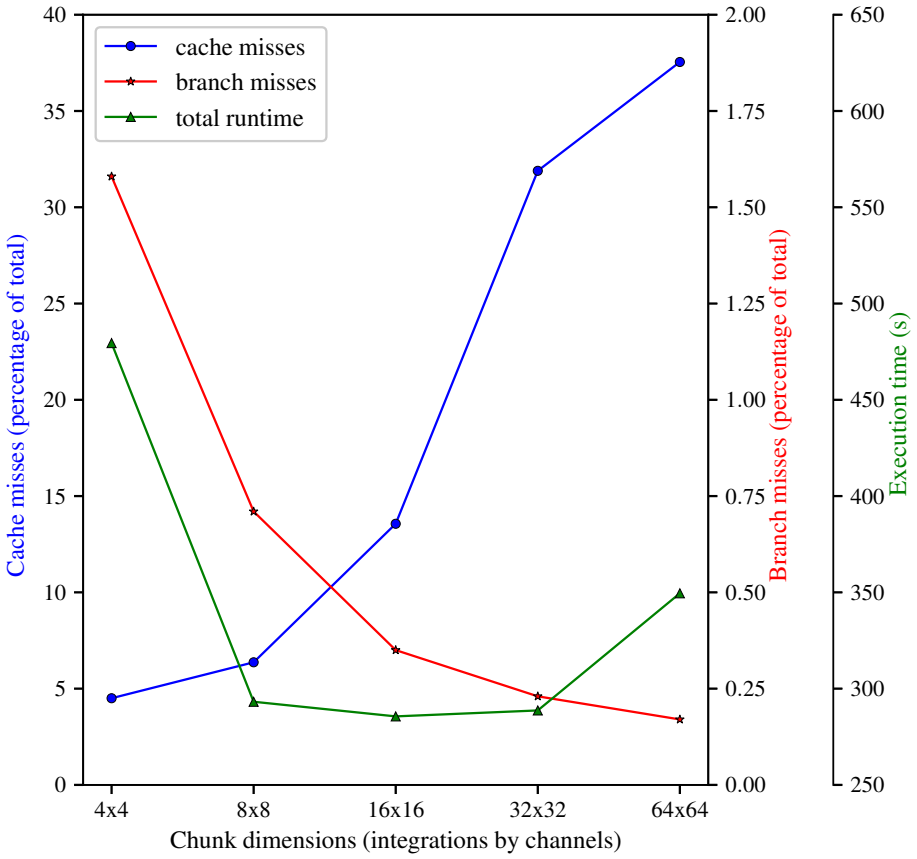

FIGURE 9.3: Plot of cache misses, branch misses and execution speed against chunk dimensions when using a single process. This highlights some of interacting factors which contribute to performance.

making use of perf for cache misses, we decided to include branch misses to see if they also impact CubiCal's performance in a meaningful way. We used the same data and once again solved for a single, direction-independent **G** term.

The results of this experiment appear in Figure 9.3. We have plotted three different quantities (cache misses, branch misses and execution time) as a function of chunk dimensions. Note that we only use a single process for these experiments - low-level profiling becomes incredibly difficult when using multi-processing. Additionally, we have normalised the execution times to correspond to a 3.10GHz core speed.

There are a few interesting features in this plot. We have similar, decent performance for (8,8), (16,16) and (32,32) time-frequency chunk dimensions. We also see that, as expected, cache misses increase dramatically as our chunk dimensions grow. This is somewhat countered by branch misses, which decrease with increasing chunk dimensions. However, the total percentage of branch misses remains much lower than that of the cache misses. Interpreting this is quite challenging, as it is clear that we can still perform well even with a relatively large percentage of cache misses as can be seen with the (32,32) data point. We also see that we can have poor performance even when we have almost no cache misses as can be seen for the (4,4) data point.

For the small chunk dimensions, we believe that we are dominated by overhead and failing

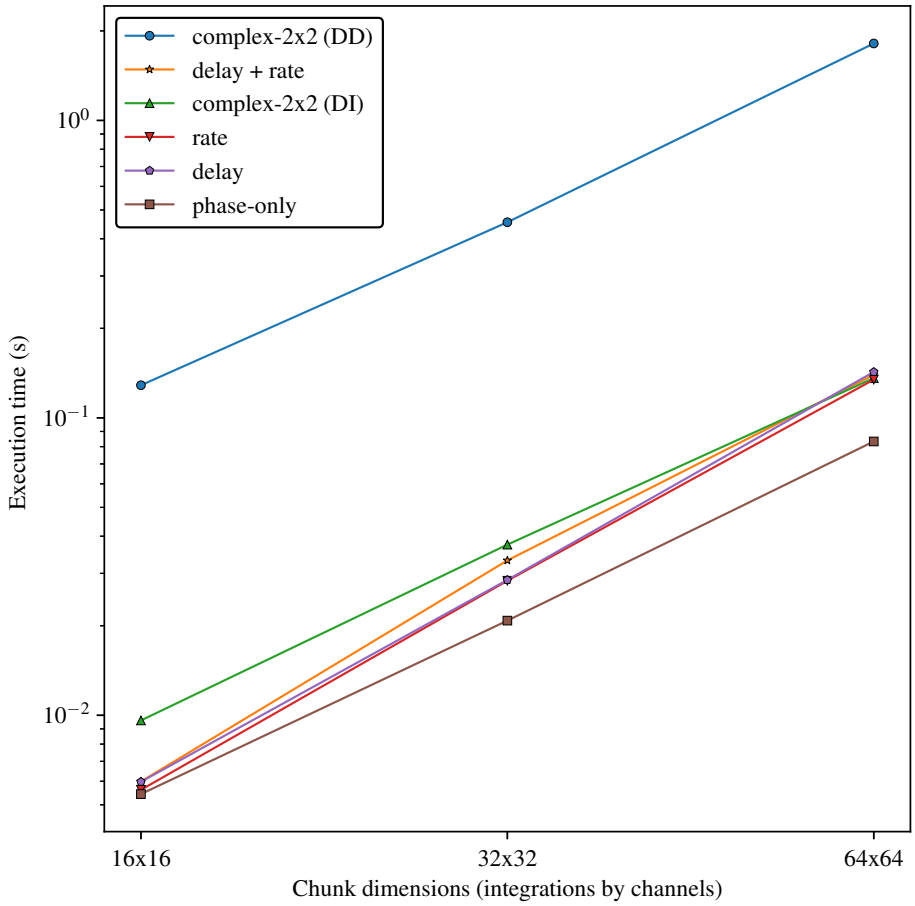

FIGURE 9.4: Plot of average per-iteration execution time against chunk dimensions for the different gain types. The straightness of the lines shows that the speed per iteration scales (approximately) linearly with chunk size.

to leverage the full power of the CPU. For the large chunk dimensions, we believe that the cache misses do account, at least partially, for the degradation in performance, but there may be additional causes which we have not yet identified. Regardless, there is certainly a hardware dependent "sweet spot" for performance.

9.4 Experiment 3

We also wanted to compare the per-iteration execution times for each of CubiCal's solvers as a function of the chunk dimensions. To do so, we calibrated the same VLA measurement set several times, using each type of gain implemented in CubiCal and three different chunk dimensions. We kept all other parameters the same (except for the direction-dependent case), and settled on a solution interval of 32 integrations (160s) and 32 channels (128MHz). This was done to ensure that gains with non-optional solution intervals (delay/rate), would function correctly. For each gain type, we averaged over the number of iterations taken to reach convergence to obtain the per-iteration times presented in Table 9.1. Figure 9.4 is a plot of the same results.

Average per-iteration execution times in milliseconds (ms)			
	Chunk dimensions (integrations by channels)		
Type	16-by-16	32-by-32	64-by-64
complex-2x2 (DI)	9.6	37.5	136.0
complex-2x2 (DD)	128.7	454.7	1814.4
phase-only	5.4	20.8	83.3
delay	6.0	28.5	142.7
rate	5.6	28.3	134.4
delay + rate	6.0	33.2	139.6

TABLE 9.1: Average per-iteration execution times for the different gains implemented in CubiCal.

It is clear from both Table 9.1 and Figure 9.4 that the per-iteration times scale linearly with chunk size. In fact, for complex-2x2 (DI), complex-2x2 (DD) and the phase-only solvers, the scaling is almost perfect - increasing the chunk size by a factor of 4 increases the execution time by a factor of 4. For the slope-based parameterised solvers, the scaling is linear but, as can be seen in Figure 9.4, the gradient of their lines is slightly steeper. We attribute this slightly poorer scaling to the more complicated inverse which is necessary in the parameterised gain updates. However, these solvers typically take very few iterations to converge.

One particularly surprising fact is that the scaling of the direction-dependent term is so well-behaved, given that it is processing a substantially larger (a factor of N_{D}) amount of data than the direction-independent terms. This does lead us to believe that the underlying Cython code is doing an excellent job.

9.5 Experiment 4

RAM is often the limiting factor when it comes to processing large data sets, as applications which exceed the available RAM end up using a swap space or a page file. These allow the system to "fake" additional memory by copying the contents of RAM to disk. However, this is very inefficient and can significantly degrade performance.

In order to provide some insight into CubiCal's memory requirements, we produced a plot its maximum memory usage as a function of data elements per tile for increasing numbers of processes (see Figure 9.5). Note that we define:

$$\text{data elements per tile} = N_t N_\nu N_A^2 N_{chunks}, \tag{9.1}$$

where N_t is the number of integrations per chunk, N_ν is the number of channels per chunk, N_A is the number of antennas and N_{chunks} is the number of chunks per tile. Recall that a tile is the larger of CubiCal's chunking objects.

Memory usage was measured using the aptly named memory_profiler[3] Python module which records memory usage as a function of time. We profiled several calibration runs on the VLA data, adjusting both the number of processes and the number of elements per tile.

[3] https://pypi.org/project/memory_profiler/

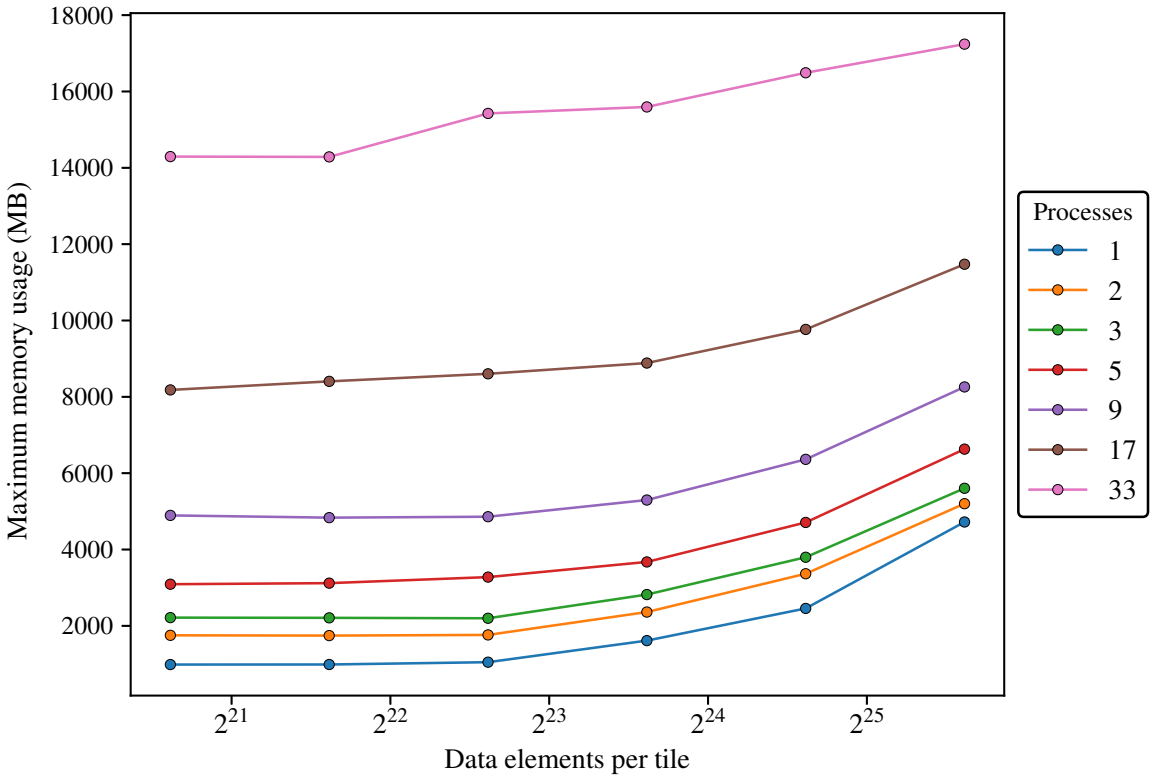

FIGURE 9.5: Plot of maximum memory usage against data elements per tile for different numbers of processes.

For this experiment, we controlled the number of elements per tile by adjusting the number of chunks per tile while keeping the time and frequency dimensions of the chunks fixed at (16,16).

Examining Figure 9.5 we can see that the maximum memory usage is quite stable as we increase the number of data elements in a single tile. Note that each data point on a single curve corresponds to a doubling of the number of elements in the tile.

The increase in memory usage with the number of processes is also consistent with what we would expect. Each process has an associated memory footprint. Consequently, if we double the number of processes, we should approximately double the maximum memory usage. At this point it is worth mentioning that the odd numbers of processes are the result of including the I/O process in the total. This was done to highlight that the blue curve has no multiprocessing (and consequently no I/O process) and serves as our baseline.

Using Figure 9.5 it is possible to predict (or extrapolate) CubiCal's maximum memory usage for a variety of problem sizes although it is worth mentioning that the average memory usage will be slightly lower. This is due to the fact that the time spent at maximum memory usage is relatively small compared to the overall execution time.

In order to better understand the memory usage within a single calibration run, we performed additional memory profiling, exploiting the functionality of memory_profiler to track the memory usage of child processes.

Figure 9.6 contains two plots of CubiCal's memory usage as a function of time. These

plots correspond to the case where we have two calibration processes, an I/O process and the main process. Note that the main process is largely idle and primarily handles process creation. The first plot of the figure is for (8,8) chunk dimensions and the second plot is for (32,32) chunk dimensions.

Comparison of the two plots reveals some interesting features. First and foremost are the large spikes in the I/O process for the (32,32) case. There are similar spikes in the (8,8) case, but they are considerably smaller. These spikes correspond to the points of maximum memory usage. They occur when we read a measurement set column and populate CubiCal's data structure: during this brief period, we have to hold two copies of the data in memory. Once CubiCal's data structure is populated, we no longer need two copies of the data and the memory usage levels out.

The sudden increase in memory usage in the (8,8) case as it nears the end of the calibration run is not interesting - it is simply the result of post-processing overhead and the creation of some diagnostic plots. In the (32,32) case, it is dwarfed by the actual I/O.

Another interesting feature is the behaviour of the calibration (labelled as children) processes. In both cases, the calibration processes behave very consistently. This is encouraging as it suggests that they are both providing equal throughput.

We have also included the results of performing the same experiment with four calibration processes (Figure 9.7). We did this to ensure that our results were consistent for a larger number of calibration processes and we were pleased to see that all the features found in the two calibration process case were also present (and slightly exaggerated) in the four calibration process case.

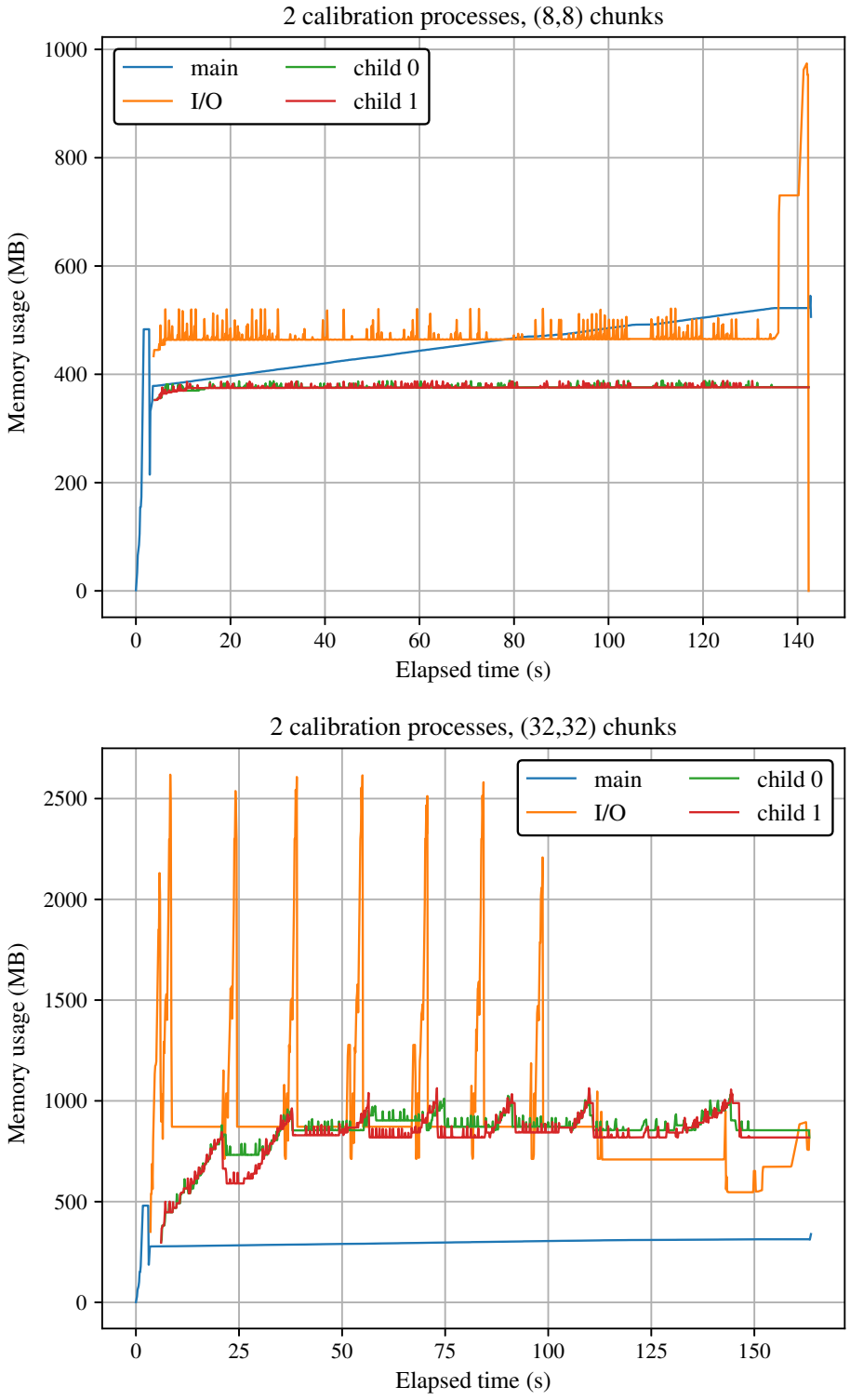

FIGURE 9.6: Plots of per-process memory usage against elapsed time when using two calibration processes. Each plot corresponds to different chunk dimensions.

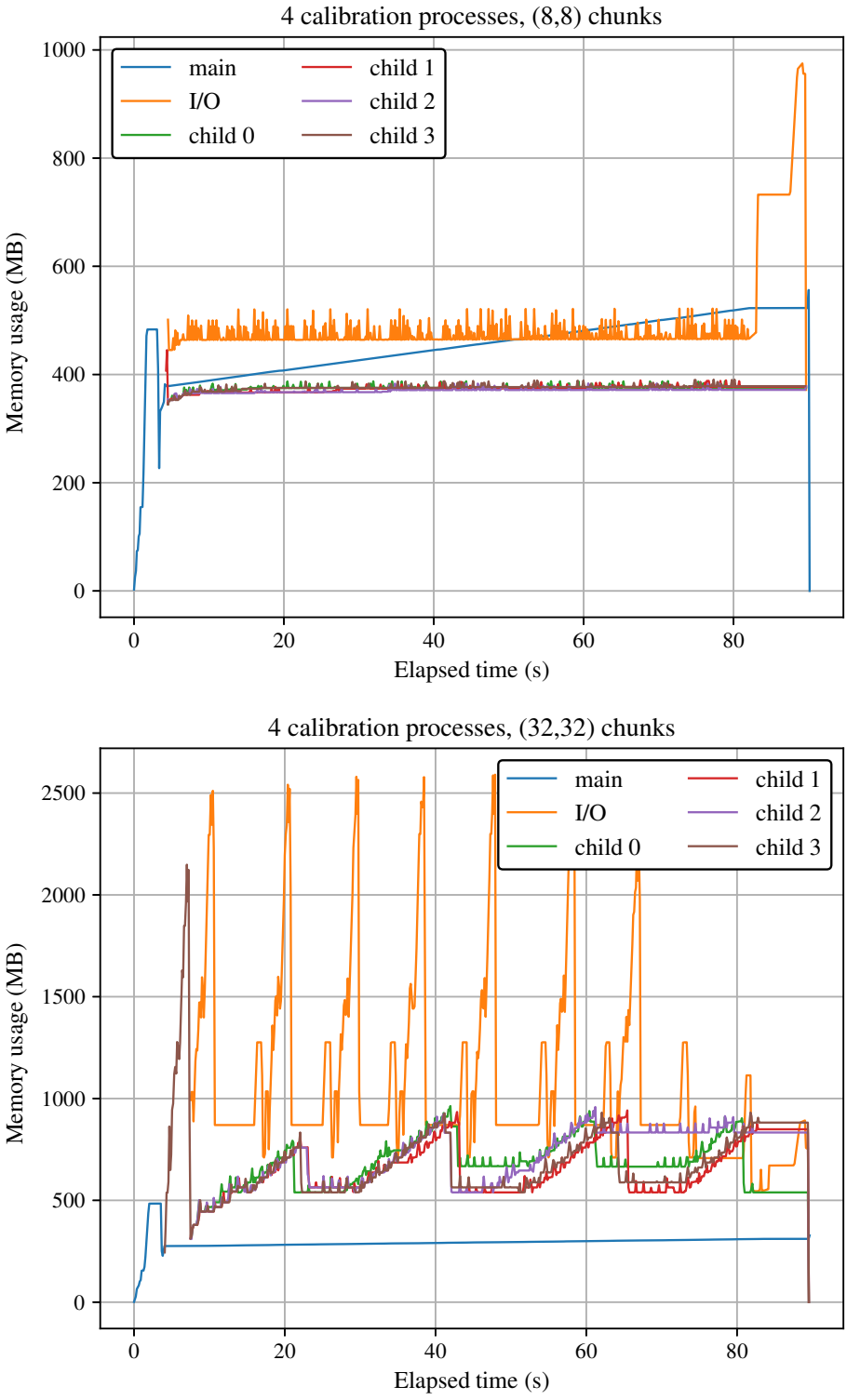

FIGURE 9.7: Plots of per-process memory usage against elapsed time when using four calibration processes. Each plot corresponds to different chunk dimensions.

Chapter 10

Conclusions and Future Work

We have presented CubiCal, a suite of fast calibration routines implemented in Python. In doing so, we have shown that applying complex optimisation to the calibration problem allows us to develop both more general and highly specialised gain solvers (Chapter 5).

The more general case includes our extension of existing work on the subject to chains of Jones terms. These Jones chains can be used to express calibration problems involving several complex gain terms succinctly, and the derived update rules avoid the usual problem of having to manipulate the observed data.

On the specialised front, we applied the ideas of complex optimisation to parameterised gains. This allowed us to derive several specialised solvers in addition to a generic method for constructing solvers for relatively arbitrary, real parametrisations of the gain. The specific specialised solvers we considered were tailored to true phase-only calibration, delay/rate calibration, and pointing error calibration. We exploited the same properties found in the more general case, such as the diagonal approximation of the Hessian, to accelerate the specialised solvers.

We discussed the specifics of CubiCal's implementation (Chapter 6), including how it has been optimised using a combination of Cython, multi-processing, intelligent data structures and a tiered data chunking strategy. Our highly modular design will allow the package to be augmented with additional solvers as they become available.

CubiCal has been successfully applied to both simulated (Chapter 7) and real (Chapter 8) visibility data. This success was achieved for several different varieties of calibration, including generic complex terms (direction-independent and direction-dependent), Jones chains and the specialised solvers. Additionally, CubiCal's small but growing user-base have obtained some excellent results. The code is already freely available, and we believe that CubiCal has the capacity to become the de facto tool for self-calibration as data volumes continue to grow.

Finally, we demonstrated that CubiCal outperforms its most comparable competitor by a substantial margin (Chapter 9), and boasts excellent scaling on parallel architectures. Some factors which impact CubiCal's performance were also analysed to provide insight into optimal input parameters.

In summary, CubiCal is a modern tool for performing radio interferometric gain calibration in the era of the SKA. Whilst it is already a competitive implementation of a new technique, there are still several directions in which CubiCal can evolve.

First and foremost, although CubiCal boasts excellent use of parallelism, it is not ready for use on distributed systems. This is fast becoming mandatory for any software tackling a big data problem. Fortunately, with packages such as Dask[1] becoming more popular and easier to use, it should be relatively simple to develop a distributed version of CubiCal. Similarly, the calibration component of CubiCal is currently CPU-based. Montblanc already has the capacity to predict visibilities using the GPU and marrying both prediction and calibration into a single GPU-based application could be tremendously efficient.

As has been mentioned previously, CubiCal's modularity makes the addition of new solvers quite simple. This includes user-developed solvers, one of which has already been incorporated (robust calibration, courtesy of U. Mbou Sob, in prep). There are also several more difficult and interesting problems which can be tackled, using an appropriate specialised solver, such as solving for Faraday Rotation. This is known to be particularly challenging to model. We also hope to include more sophisticated solvers which will obviate the omnipresent solution interval problem (Bester et al., in prep). This will likely be accomplished by incorporating advanced regularisation techniques in order to improve the quality of the gain solutions. Additionally, we need to give further thought to solvers which require numerical differentiation (such as pointing error) and to the modification of existing solvers. One such modification would be the adaptation of the delay solver to model delays proportional to the inverse of the frequency as opposed to the frequency itself. This is consistent with delays introduced by the ionosphere.

There is also the potential for further optimisation of the existing implementation. CubiCal is currently very reliant on Cython, but Numba[2], a just-in-time compiler for Python, has matured substantially since CubiCal's inception. We aim to test whether it can outperform our existing low-level code in the near future.

It is also important to recall that CubiCal is not currently an end-to-end solution - it is reliant on an external imager for generating and updating sky models. This is functionality which would be worth incorporating into CubiCal from an optimisation perspective and in order to make solving for source parameters tractable. Perhaps the first step in this direction would be the addition of a (facet-based) degridder for model prediction. This would also go some way to addressing the difficult problem of constructing direction-dependent models.

On that note, we must draw attention to the fact that CubiCal's direction-dependent calibration is currently best suited to the finite direction case. It performs well for solving towards a relatively small number of the dynamic range limiting bright sources, but it cannot accurately correct every source in the field of view. Parameterised solvers which can address this problem are already being investigated.

From a more mathematical perspective, there is the difficulty of including a parameterised gain in a Jones chain whilst using the 2×2 formalism. This can be addressed by reformulating the Jones chain mathematics using the more general 4×4 matrix formalism. Naturally, this is a fairly large paradigm shift which is not in keeping with CubiCal's original design. However, tentative plans are in place for a revised version of CubiCal which will be entirely 4×4 matrix based and boast superior flexibility.

[1] https://dask.org/
[2] http://numba.pydata.org/

There is one final omission from this document: the absence of an explicit polarisation experiment. Whilst this may seem peculiar given the emphasis placed on our full-polarisation treatment of the Jones chain, it is largely a practical limitation. Polarisation experiments are particularly challenging due to their reliance on multiple calibrator observations and the relative lack of good polarisation data. That being said, several users have already used CubiCal for polarisation calibration and report excellent results. This functionality requires further investigation but is certainly a promising start given how different our approach is to that taken by tools such as CASA.

Appendix A

Software

CubiCal's code base (with the exception of contributions by collaborators) and documentation are an integral part of this manuscript and should be regarded as digital appendices. For the sake of convenience, we will reiterate that CubiCal is available at:

https://github.com/ratt-ru/CubiCal/.

The documentation is available at:

https://cubical.readthedocs.io/.

A.1 Installation

Whilst all the installation instructions are available in the documentation, we will reproduce the instructions for installation on Ubuntu 18.04 LTS here. First and foremost, some dependencies need to be installed from KERN[1], a suite of easy-to-install software packages for radio astronomy:

```
apt-get install software-properties-common
apt-add-repository -s ppa:kernsuite/kern-4
apt-add-repository multiverse
apt-add-repository restricted
apt-get update
apt-get install -y casacore-dev libboost-python-dev
apt-get install -y libcfitsio3-dev wcslib-dev
```

Note that CubiCal itself will be included in the next release of KERN. Having satisfied the dependencies, CubiCal can be installed directly from the release branch on GitHub:

```
pip install git+https://github.com/ratt-ru/CubiCal.git@release-branch
```

Developers are encouraged to install from source, the instructions for which are in the documentation. Note that support of component-based sky models (as opposed to model visibility columns) requires the installation of Montblanc[2].

[1] http://kernsuite.info/
[2] https://montblanc.readthedocs.io/

Bibliography

Balanis, C. A. (2005). *Antenna theory: analysis and design*. Wiley-Interscience.

Baldwin, J. E. and P. J. Warner (1978). "Phaseless aperture synthesis". In: *MNRAS* 182, pp. 411–422. DOI: 10.1093/mnras/182.3.411.

Behnel, S. et al. (2011). "Cython: The Best of Both Worlds". In: *Computing in Science Engineering* 13.2, pp. 31 –39. ISSN: 1521-9615. DOI: 10.1109/MCSE.2010.118.

Bhatnagar, S. and T. J. Cornwell (2017). "The Pointing Self-calibration Algorithm for Aperture Synthesis Radio Telescopes". In: *AJ* 154, p. 197. DOI: 10.3847/1538-3881/aa8f43. arXiv: 1709.08681 [astro-ph.IM].

Bhatnagar, S. and R. Nityananda (2001). "Solving for closure errors due to polarization leakage in radio interferometry of unpolarized sources". In: *A&A* 375, pp. 344–350. DOI: 10.1051/0004-6361:20010799. eprint: astro-ph/0106348.

Bhatnagar, S. et al. (2004). "Solving for the antenna based pointing errors". In: *EVLA Memo# 84, National Radio Astronomy Observatory*.

Bhatnagar, S. et al. (2008). "Correcting direction-dependent gains in the deconvolution of radio interferometric images". In: *A&A* 487, pp. 419–429. DOI: 10.1051/0004-6361:20079284. arXiv: 0805.0834.

Bhatnagar, S. et al. (2013). "Wide-field wide-band Interferometric Imaging: The WB A-Projection and Hybrid Algorithms". In: *ApJ* 770, p. 91. DOI: 10.1088/0004-637X/770/2/91. arXiv: 1304.4987 [astro-ph.IM].

Born, M. and E. Wolf (2013). *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. Elsevier.

Boyd, S. et al. (2011). "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine learning* 3.1, pp. 1–122.

Brisken, W. (2003). *EVLA Memo 58. Using Grasp8 To Study The VLA Beam*.

Candes, E. et al. (2005). "Stable Signal Recovery from Incomplete and Inaccurate Measurements". In: *ArXiv Mathematics e-prints*. eprint: math/0503066.

Chiarucci, S. and S. J. Wijnholds (2018). "Blind calibration of radio interferometric arrays using sparsity constraints and its implications for self-calibration". In: *MNRAS* 474, pp. 1028–1040. DOI: 10.1093/mnras/stx2731.

Cornwell, T. et al. (2011). *ASKAP Science Processing*. Tech. rep. ASKAP-SW-0020.

Cornwell, T. J. and P. N. Wilkinson (1981). "A new method for making maps with unstable radio interferometers". In: *MNRAS* 196, pp. 1067–1086. DOI: 10.1093/mnras/196.4.1067.

Cotton, W. D. (1979). "A method of mapping compact structure in radio sources using VLBI observations". In: *AJ* 84, pp. 1122–1128. DOI: 10.1086/112519.

— (1995). "Fringe Fitting". In: *Very Long Baseline Interferometry and the VLBA*. Ed. by J. A. Zensus et al. Vol. 82. Astronomical Society of the Pacific Conference Series, p. 189.

— (2007). *EVLA Memo 118. Ionospheric Effects and Imaging and Calibration of VLA Data*.

Dempster A, P. et al. (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38.

Donoho, D. L. (2006). "Compressed sensing". In: *IEEE Transactions on information theory* 52.4, pp. 1289–1306.

Ekers, R. D. (1984). "The almost Serendipitous Discovery of Self-Calibration". In: *Serendipitous Discoveries in Radio Astronomy*. Ed. by K. I. Kellermann and B. Sheets, p. 154.

Enßlin, T. A. et al. (2014). "Improving self-calibration". In: *Phys. Rev. E* 90.4. DOI: 10.1103/PhysRevE.90.043301. arXiv: 1312.1349 [astro-ph.IM].

Fessler, J. A and A. O Hero (1994). "Space-alternating generalized expectation-maximization algorithm". In: *IEEE Transactions on Signal Processing* 42.10, pp. 2664–2677.

Fomalont, E. B. and R. A. Perley (1999). "Calibration and Editing". In: *Synthesis Imaging in Radio Astronomy II*. Ed. by G. B. Taylor et al. Vol. 180. Astronomical Society of the Pacific Conference Series, p. 79.

Fort, D. N. and H. K. C. Yee (1976). "A Method of Obtaining Brightness Distributions from Long Baseline Interferometry". In: *A&A* 50, p. 19.

Friedman, J. et al. (2007). "Pathwise coordinate optimization". In: *The Annals of Applied Statistics* 1.2, pp. 302–332.

Graf, R. F. (1999). *Modern dictionary of electronics*. Elsevier.

Greisen, E. W. (2016). *The AIPS FAQ*. URL: http://www.aips.nrao.edu/aips_faq.html (visited on 07/24/2018).

Greisen, E. W. and A. Bridle (1985). *AIPS Cookbook*. National Radio Astronomy Observatory.

Grobler, T. L. et al. (2014). "Calibration artefacts in radio interferometry - I. Ghost sources in Westerbork Synthesis Radio Telescope data". In: *MNRAS* 439, pp. 4030–4047. DOI: 10.1093/mnras/stu268. arXiv: 1402.1373 [astro-ph.IM].

Grobler, T. L. et al. (2016). "Calibration artefacts in radio interferometry - III. Phase-only calibration and primary beam correction". In: *MNRAS* 461, pp. 2975–2992. DOI: 10.1093/mnras/stw1437. arXiv: 1606.06320 [astro-ph.IM].

Hamaker, J. P. (2000). "Understanding radio polarimetry. IV. The full-coherency analogue of scalar self-calibration: Self-alignment, dynamic range and polarimetric fidelity". In: *A&AS* 143, pp. 515–534. DOI: 10.1051/aas:2000337.

Hamaker, J. P. et al. (1996). "Understanding radio polarimetry. I. Mathematical foundations." In: *A&AS* 117, pp. 137–147.

Högbom, J. A. (1974). "Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines". In: *A&AS* 15, p. 417.

Hudson, H. M. and R. S. Larkin (1994). "Accelerated image reconstruction using ordered subsets of projection data". In: *IEEE transactions on medical imaging* 13.4, pp. 601–609.

Intema, H. T. et al. (2009). "Ionospheric calibration of low frequency radio interferometric observations using the peeling scheme. I. Method description and first results". In: *A&A* 501, pp. 1185–1205. DOI: 10.1051/0004-6361/200811094. arXiv: 0904.3975 [astro-ph.IM].

Jansky, K. G. (1932). "Directional studies of atmospherics at high frequencies". In: *Classics in Radio Astronomy*. Springer, pp. 10–22.

— (1933). "Electrical disturbances apparently of extraterrestrial origin". In: *Proceedings of the Institute of Radio Engineers* 21.10, pp. 1387–1398.

Jennison, R. C. (1954). "Measurement of the fine structure of the cosmic radio forces". PhD thesis. University of Manchester.

— (1958). "A phase sensitive interferometer technique for the measurement of the Fourier transforms of spatial brightness distributions of small angular extent". In: *MNRAS* 118, p. 276. DOI: 10.1093/mnras/118.3.276.

Jones, E. et al. (2001–). *SciPy: Open source scientific tools for Python*. URL: http://www.scipy.org/ (visited on 08/29/2018).

Jones, R. C. (1941). "New calculus for the treatment of optical systems. I. Description and discussion of the calculus". In: *Journal of the Optical Society of America (1917-1983)* 31, p. 488.

Kalman, R. E. (1960). "A new approach to linear filtering and prediction problems". In: *Journal of basic Engineering* 82.1, pp. 35–45.

Kazemi, S. and S. Yatawatta (2013). "Robust radio interferometric calibration using the t-distribution". In: *MNRAS* 435, pp. 597–605. DOI: 10.1093/mnras/stt1347. arXiv: 1307.5040 [astro-ph.IM].

Kazemi, S. et al. (2011). "Radio interferometric calibration using the SAGE algorithm". In: *MNRAS* 414, pp. 1656–1666. DOI: 10.1111/j.1365-2966.2011.18506.x. arXiv: 1012.1722 [astro-ph.IM].

Kazemi, S. et al. (2013). "Radio interferometric calibration via ordered-subsets algorithms: OS-LS and OS-SAGE calibrations". In: *MNRAS* 434, pp. 3130–3141. DOI: 10.1093/mnras/stt1229. arXiv: 1307.0125 [astro-ph.IM].

Kazemi, S. et al. (2015). "Blind calibration for radio interferometry using convex optimization". In: *Compressed Sensing Theory and its Applications to Radar, Sonar and Remote Sensing (CoSeRa), 2015 3rd International Workshop on*. IEEE, pp. 164–168.

Kenyon, J. S. et al. (2018). "CubiCal - fast radio interferometric calibration suite exploiting complex optimization". In: *MNRAS* 478, pp. 2399–2415. DOI: 10.1093/mnras/sty1221. arXiv: 1805.03410 [astro-ph.IM].

Kreutz-Delgado, K. (2009). "The Complex Gradient Operator and the CR-Calculus". In: *ArXiv e-prints*. arXiv: 0906.4835 [math.OC].

Levenberg, K. (1944). "A method for the solution of certain non-linear problems in least squares". In: *Quarterly of applied mathematics* 2.2, pp. 164–168.

Liu, C. and D. B. Rubin (1995). "ML estimation of the t distribution using EM and its extensions, ECM and ECME". In: *Statistica Sinica*, pp. 19–39.

Lonsdale, C. J. (2005). "Configuration Considerations for Low Frequency Arrays". In: *From Clark Lake to the Long Wavelength Array: Bill Erickson's Radio Science*. Vol. 345, p. 399.

Madsen, K. et al. (2004). *Methods for Non-Linear Least Squares Problems (2nd ed.)* eng.

Makhathini, S. (2018). "Advanced radio interferometric simulation and data reduction techniques". PhD thesis. Rhodes University.

Marquardt, D. W. (1963). "An algorithm for least-squares estimation of nonlinear parameters". In: *Journal of the society for Industrial and Applied Mathematics* 11.2, pp. 431–441.

McMullin, J. P. et al. (2007). "CASA Architecture and Applications". In: *Astronomical Data Analysis Software and Systems XVI*. Ed. by R. A. Shaw et al. Vol. 376. Astronomical Society of the Pacific Conference Series, p. 127.

Michelson, A. A. and E. W. Morley (1887). "On the Relative Motion of the Earth and of the Luminiferous Ether". In: *Sidereal Messenger* 6, pp. 306–310.

Mitchell, D. A. et al. (2008). "Real-Time Calibration of the Murchison Widefield Array". In: *IEEE Journal of Selected Topics in Signal Processing* 2, pp. 707–717. DOI: 10.1109/JSTSP.2008.2005327.

Mitra, M. et al. (2015). "Incorporation of antenna primary beam patterns in radio-interferometric data reduction to produce wide-field, high-dynamic-range images". In: *2015 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, pp. 494–497. DOI: 10.1109/ICEAA.2015.7297163.

Mohan, N. and D. Rafferty (2015). *PyBDSF: Python Blob Detection and Source Finder*. Astrophysics Source Code Library. ascl: 1502.007.

Mouri Sardarabadi, A. et al. (2016). "Radio astronomical image formation using constrained least squares and Krylov subspaces". In: *A&A* 588, A95. DOI: 10.1051/0004-6361/201526214. arXiv: 1505.01348 [astro-ph.IM].

Noordam, J. E. (1982). *The JEEVES cookbook*. URL: http://www.astron.nl/sites/astron.nl/files/cms/Documenten/Notes/ASTRON-NOTE-374.pdf.

— (1994). "The NEWSTAR cookbook". In: *Internal NFRA report*.

— (1996). *The measurement equation of a generic radio telescope, AIPS++ implementation note nr 185*.

— (2004). "LOFAR calibration challenges". In: *Ground-based Telescopes*. Ed. by J. M. Oschmann Jr. Vol. 5489. Proc. SPIE, pp. 817–825. DOI: 10.1117/12.544262.

Noordam, J. E. and A. G. de Bruyn (1982). "High dynamic range mapping of strong radio sources, with application to 3C84". In: *Nature* 299.5884, p. 597.

Noordam, J. E. and O. M. Smirnov (2010). "The MeqTrees software system and its use for third-generation calibration of radio interferometers". In: *A&A* 524, A61. DOI: 10.1051/0004-6361/201015013. arXiv: 1101.1745 [astro-ph.IM].

Offringa, A. R. et al. (2014). "WSCLEAN: an implementation of a fast, generic wide-field imager for radio astronomy". In: *MNRAS* 444, pp. 606–619. DOI: 10.1093/mnras/stu1368.

Oliphant, T. E. (2006). *A guide to NumPy*. Vol. 1. Trelgol Publishing USA.

Ollier, V. et al. (2016). "Relaxed concentrated MLE for robust calibration of radio interferometers". In: *Signal Processing Conference (EUSIPCO)*. IEEE, pp. 280–284.

— (2017). "Robust Calibration of Radio Interferometers in Non-Gaussian Environment". In: *IEEE Transactions on Signal Processing* 65.21, pp. 5649–5660.

Ott, J. and J. Kern (2017). *CASA Synthesis & Single Dish Reduction Reference Manual & Cookbook*.

Pearson, T. J. and A. C. S. Readhead (1984). "Image Formation by Self-Calibration in Radio Astronomy". In: *ARA&A* 22, pp. 97–130. DOI: 10.1146/annurev.aa.22.090184.000525.

Perkins, S. J. et al. (2015). "Montblanc: GPU accelerated radio interferometer measurement equations in support of Bayesian inference for radio observations". In: *Astronomy and Computing* 12, pp. 73–85. DOI: 10.1016/j.ascom.2015.06.003. arXiv: 1501.07719 [cs.DC].

Perley, R. (2013). "High Dynamic Range Imaging". Presentation at "The Radio Universe @ Ger's (wave)-length". URL: http://www.astron.nl/gerfeest/presentations/perley.pdf (visited on 08/29/2018).

Readhead, A. C. S. and P. N. Wilkinson (1978). "The mapping of compact radio sources from VLBI data". In: *ApJ* 223, pp. 25–36. DOI: 10.1086/156232.

Readhead, A. C. S. et al. (1980). "Mapping radio sources with uncalibrated visibility data". In: *Nature* 285, pp. 137–140. DOI: 10.1038/285137a0.

Reber, G. (1940). "Cosmic static". In: *Proceedings of the IRE* 28.2, pp. 68–70.

Repetti, A. et al. (2017). "Non-convex optimization for self-calibration of direction-dependent effects in radio interferometric imaging". In: *MNRAS* 470, pp. 3981–4006. DOI: 10.1093/mnras/stx1267. arXiv: 1701.03689 [astro-ph.IM].

Rogers, A. E. E. et al. (1974). "The structure of radio sources 3C 273B and 3C 84 deduced from the 'closure' phases and visibility amplitudes observed with three-element interferometers". In: *ApJ* 193, pp. 293–301. DOI: 10.1086/153162.

Salvini, S. and S. J. Wijnholds (2014). "Fast gain calibration in radio astronomy using alternating direction implicit methods: Analysis and applications". In: *A&A* 571, A97. DOI: 10.1051/0004-6361/201424487. arXiv: 1410.2101 [astro-ph.IM].

Sault, R. J. and N. E. B. Killeen (2004). *The Miriad User's Guide. Australia Telescope National Facility, Sydney.*

Sault, R. J. et al. (1995). "A Retrospective View of MIRIAD". In: *Astronomical Data Analysis Software and Systems IV*. Ed. by R. A. Shaw et al. Vol. 77. Astronomical Society of the Pacific Conference Series, p. 433. eprint: astro-ph/0612759.

Schwab, F. R. (1980). "Adaptive calibration of radio interferometer data". In: *1980 International Optical Computing Conference I*. Ed. by W. T. Rhodes. Vol. 231. Proc. SPIE, pp. 18–25. DOI: 10.1117/12.958828.

Shepherd, M. C. (1997). "Difmap: an Interactive Program for Synthesis Imaging". In: *Astronomical Data Analysis Software and Systems VI*. Ed. by G. Hunt and H. Payne. Vol. 125. Astronomical Society of the Pacific Conference Series, p. 77.

Shimwell, T. W. et al. (2019). "The LOFAR Two-metre Sky Survey. II. First data release". In: *A&A* 622, A1. DOI: 10.1051/0004-6361/201833559. arXiv: 1811.07926.

Smirnov, O. M. (2011a). "Revisiting the radio interferometer measurement equation. I. A full-sky Jones formalism". In: *A&A* 527, A106. DOI: 10.1051/0004-6361/201016082. arXiv: 1101.1764 [astro-ph.IM].

— (2011b). "Revisiting the radio interferometer measurement equation. II. Calibration and direction-dependent effects". In: *A&A* 527, A107. DOI: 10.1051/0004-6361/201116434. arXiv: 1101.1765 [astro-ph.IM].

— (2011c). "Revisiting the radio interferometer measurement equation. III. Addressing direction-dependent effects in 21 cm WSRT observations of 3C 147". In: *A&A* 527, A108. DOI: 10.1051/0004-6361/201116435. arXiv: 1101.1768 [astro-ph.IM].

Smirnov, O. M. and C. Tasse (2015). "Radio interferometric gain calibration as a complex optimization problem". In: *MNRAS* 449, pp. 2668–2684. DOI: 10.1093/mnras/stv418. arXiv: 1502.06974 [astro-ph.IM].

Smith, F. G. (1952). "The Measurement of the Angular Diameter of Radio Stars". In: *Proceedings of the Physical Society. Section B* 65.12, p. 971. URL: http://stacks.iop.org/0370-1301/65/i=12/a=309.

Sorber, L. et al. (2012). "Unconstrained Optimization of Real Functions in Complex Variables". In: *SIAM J. Optim.* 22.3, pp. 879–898. DOI: 10.1137/110832124. URL: https://doi.org/10.1137/110832124.

Tasse, C. (2014a). "Applying Wirtinger derivatives to the radio interferometry calibration problem". In: *ArXiv e-prints*. arXiv: 1410.8706 [astro-ph.IM].

— (2014b). "Nonlinear Kalman filters for calibration in radio interferometry". In: *A&A* 566, A127. DOI: 10.1051/0004-6361/201423503. arXiv: 1403.6308 [astro-ph.IM].

Tasse, C. et al. (2018). "Faceting for direction-dependent spectral deconvolution". In: *A&A* 611, A87. DOI: 10.1051/0004-6361/201731474. arXiv: 1712.02078 [astro-ph.IM].

Taylor, G. (1994). *The Difmap Cookbook*. California Institute of Technology, Pasadena.

The Astropy Collaboration et al. (2018). "The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package". In: *AJ* 156, p. 123. DOI: 10.3847/1538-3881/aabc4f. arXiv: 1801.02634 [astro-ph.IM].

Thompson, A. R. and L. R. D'Addario (1982). "Frequency response of a synthesis array - Performance limitations and design tolerances". In: *Radio Science* 17, pp. 357–369. DOI: 10.1029/RS017i002p00357.

Thompson, A. R. et al. (2017). *Interferometry and Synthesis in Radio Astronomy, 3rd Edition*. DOI: 10.1007/978-3-319-44431-4.

Twiss, R. Q. et al. (1960). "Brightness distribution over some strong radio sources at 1427 Mc/s". In: *The Observatory* 80, pp. 153–159.

van Cittert, P. H. (1934). "Die Wahrscheinliche Schwingungsverteilung in Einer von Einer Lichtquelle Direkt Oder Mittels Einer Linse Beleuchteten Ebene". In: *Physica* 1.1, pp. 201 –210. ISSN: 0031-8914. DOI: https://doi.org/10.1016/S0031-8914(34)90026-4. URL: http://www.sciencedirect.com/science/article/pii/S0031891434900264.

van der Hulst, J. M. et al. (1992). "The Groningen Image Processing SYstem, GIPSY". In: *Astronomical Data Analysis Software and Systems I*. Ed. by D. M. Worrall et al. Vol. 25. Astronomical Society of the Pacific Conference Series, p. 131.

van Weeren, R. J. et al. (2016). "LOFAR Facet Calibration". In: *ApJS* 223, p. 2. DOI: 10.3847/0067-0049/223/1/2. arXiv: 1601.05422 [astro-ph.IM].

Wijnholds, S. J. et al. (2016). "Calibration artefacts in radio interferometry - II. Ghost patterns for irregular arrays". In: *MNRAS* 457, pp. 2331–2354. DOI: 10.1093/mnras/stw118.

Wilson, T. L. et al. (2009). *Tools of radio astronomy*. Vol. 5. Springer.

Wirtinger, W. (1927). "Zur formalen Theorie der Funktionen von mehr komplexen Veränderlichen". In: *Mathematische Annalen* 97.1, pp. 357–375. ISSN: 1432-1807. DOI: 10.1007/BF01447872.

Yatawatta, S. (2013). "Radio Interferometric Calibration Using a Riemannian Manifold". In: *ArXiv e-prints*. arXiv: 1303.1029 [astro-ph.IM].

— (2015). "Distributed radio interferometric calibration". In: *MNRAS* 449, pp. 4506–4514. DOI: 10.1093/mnras/stv596. arXiv: 1502.00858 [astro-ph.IM].

Yatawatta, S. and S. Kazemi (2014). "Robust radio interferometric calibration". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, pp. 5392–5396.

Yatawatta, S. et al. (2008). "Radio Interferometric Calibration Using The SAGE Algorithm". In: *ArXiv e-prints*. arXiv: 0810.5751.

Yatawatta, S. et al. (2017). "Adaptive ADMM in Distributed Radio Interferometric Calibration". In: *ArXiv e-prints*. arXiv: 1710.05656 [astro-ph.IM].

Yatawatta, S. et al. (2018). "Data multiplexing in radio interferometric calibration". In: *MNRAS* 475, pp. 708–715. DOI: 10.1093/mnras/stx3130. arXiv: 1711.10221 [astro-ph.IM].

Zernike, F. (1938). "The concept of degree of coherence and its application to optical problems". In: *Physica* 5.8, pp. 785 –795. ISSN: 0031-8914. DOI: https://doi.org/10.1016/S0031-8914(38)80203-2. URL: http://www.sciencedirect.com/science/article/pii/S0031891438802032.