

UNIVERSITY OF GRONINGEN

ASTRONOMY BACHELOR THESIS

Directional calibration of LOFAR

Using Statistically Efficient and Fast Calibration

Author:
Nikki ARENDSE

Supervisors:
Tammo Jan DIJKEMA
Dr. John MCKEAN



July 13, 2015

Abstract

Interferometers such as LOFAR provide a way to increase the diameter of radio telescopes, hence increasing the angular resolution. Since the radio waves enter the ionosphere at different positions, they experience a different phase shift. To produce one consistent image all the data from the stations needs to be calibrated for this phase shift. In the measurement equation the corruption of the data is represented by a gain matrix, and solving for this matrix gives a quadratic matrix equation. The approach used in my thesis to approximate the true value of the gain matrix is inspired by the work of Cyril Tasse and is a direction dependent extension of 'StEFCal': Statistically Efficient and Fast Calibration. The method used is Alternating Direction Implicit, combined with a relaxation step after every second iteration. The aim of this thesis is to explain the underlying principles of interferometry and directional calibration, provide an algorithm that can successfully calibrate the data of LOFAR and evaluate the effects of different frequencies, numbers of stations and numbers of directions on the convergence of the gain matrices.

Contents

1	Astronomical background	3
1.1	Introduction	3
1.2	Radio astronomy	3
1.3	Interferometry and visibilities	4
1.4	Independent closure phases	6
1.5	Ionosphere	7
1.6	Measurement equation	7
2	Calibration	9
2.1	Introduction	9
2.2	Non-directional calibration	10
2.3	Multiple data sets per station	11
2.4	Directional calibration	12
3	Experimental results	15
3.1	Variable number of directions	15
3.2	Convergence plots	16
3.2.1	Dependence on number of directions	16
3.2.2	Dependence on number of stations	18
3.2.3	Averaging the frequencies	18
3.2.4	Without the relaxation process	18
3.2.5	Individual gain elements	19
4	Discussion and future research	21
5	Conclusion	21
6	Acknowledgements	23
7	Appendix	24
7.1	Solving over-determined systems	24
7.2	Python scripts	26

1 Astronomical background

1.1 Introduction

Astronomy is our view on the Universe and everything in it. By collecting light from the sky humans are discovering more and more about the processes that drive stars, the extreme conditions near black holes and pulsars, the expansion of the Universe and much more unanswered questions. Radio astronomy plays an important role in this process, since a lot of the light from space is emitted in the radio regime. Radio interferometers such as LOFAR consist of multiple telescopes combined to accomplish a large diameter. An unresolved challenge is the calibration of all the different signals coming in from the telescopes.

In this thesis a method of direction dependent calibration is evaluated to correct the incoming data. This chapter explains the theory behind radio interferometers, describes the influence of the ionosphere and introduces the measurement equation.

1.2 Radio astronomy

Let us start with a brief overview of the history of radio astronomy, which started with James Maxwell and Heinrich Hertz in the 19th century. Maxwell discovered that electrical fields and magnetic fields can couple together to form electromagnetic waves, and proposed his four famous equations to describe how these waves propagate. The speed that followed from these equations was equal to the speed of light. In 'A Dynamical Theory of the Electromagnetic Field', he commented:

"The agreement of the results seems to show that light and magnetism are affections of the same substance, and that light is an electromagnetic disturbance propagated through the field according to electromagnetic laws."

His theory predicted that light should exist at any wavelength, not only in the visible range. Hereby Maxwell had in theory described radio waves, but not yet proven that they did indeed exist. Heinrich Hertz provided the solution, when he created a radio wave by using two rods as a receiver and a spark gap as the receiving antenna.

For years people unsuccessfully tried to detect radio waves from space, until in 1930 Karl Jansky accidentally discovered radio waves from the

densest part of the Milky Way. It took some time before astronomers started to see the importance of radio astronomy. At first they were hesitant, because of technical difficulties and because they didn't immediately see the use of it.

Stars are the main source of radiation in the sky, and even the coolest stars send most of their radiation in the visible and infra-red spectra, leaving only a small amount in the radio regime. However, radio radiation turned out to be of great importance in several different aspects of astronomy.

Cold gas clouds in the interstellar medium emit radio waves in a wide variety of molecular lines, including the 21-cm line which is due to the spin-flip in hydrogen atoms. Another example of thermal radiation is the background radiation from the CMB, which acts like a black body with a temperature of 2.7 K. Most of the photons emitted by the CMB have a wavelength of approximately 1 mm, making them detectable by radio telescopes. These photons are the oldest light in our Universe, and can tell us more about its beginning and the structure we see today.

Using radio astronomy, both thermal and non-thermal radiation can be detected [Burke and Graham-Smith, 2002]. Thermal radiation is the radiation emitted by a source solely due to its temperature, the frequency spectrum resembles more or less that of a black-body spectrum. Non-thermal radiation is all the radiation due to a different source, such as synchrotron radiation and the inverse Compton process. Synchrotron radiation is emitted by high-energy particles with relativistic velocities going through a magnetic field. This type of radiation can be seen in the most energetic objects in the Universe, such as quasars, neutron-stars and black holes. Furthermore, radio telescopes can capture light that has been redshifted during its long journey through space, originating from old times such as the epoch of re-ionization. All these different applications of radio astronomy can tell us more about many astrophysical aspects, for this purpose many radio telescopes are being built all over the world. One of the challenges of radio astronomy is to achieve a sufficiently large telescope diameter. This chapter will explain the concept of interferometry, which provides a way of increasing the diameter by combining multiple small telescopes.

1.3 Interferometry and visibilities

In this section the concept of interferometry and visibilities will be explained. Interferometers are needed to increase the angular resolution of a telescope, which is the ability to distinguish small objects. The angular resolution is given by

$$\theta = \frac{\lambda}{D} \quad (1.1)$$

in which θ is given in radians, λ corresponds to the wavelength of the light and D equals the diameter of the telescope. Radio waves have a very large wavelength, which means that the diameter of the telescope needs to be incredibly large to get sub-arcsecond resolution. The largest aperture radio telescope in the world is located in Puerto Rico, and has a diameter of 305 meters. However, for radio waves with a wavelength of 1 meter, this still only gives an angular resolution of $\theta = 676$ arcseconds. Interferometry gives the solution to this problem, and combines many smaller radio telescopes to attain a higher angular resolution, as was discovered by Joseph Lade Pawsey and Ruby Payne-Scott in 1946.

An interferometer consists of N antennas, which all measure incoming radio waves of a certain frequency and transform them to a voltage. Interferometry makes use of the wave-like properties of light. Let us first look at the case when there are only two stations. Two radio waves come in at a certain angle, which determines the difference between them. Depending on this difference, they can either add to each other when they are in phase, or cancel each other when they are out of phase. [*Fringe Dwellers*, 2015]. The output of the combined waves gives a fringe pattern, which is a pattern of evenly spaced alternating bright and dark bands, as can be seen in Figure 2.

In Figure 1 two antennas and the incoming radio waves are shown. The source of the radio waves can be assumed to be far away, hence the radio waves appear to come in parallel to each other. The stations are sensitive to radio waves of a very small frequency range, centered around $\nu = \frac{\omega}{2\pi}$. We want to compute the response between the two stations, for which we first need to multiply the voltages coming out from the stations. These voltages depend on the delay time τ_g , the frequency and the time, and can be represented by

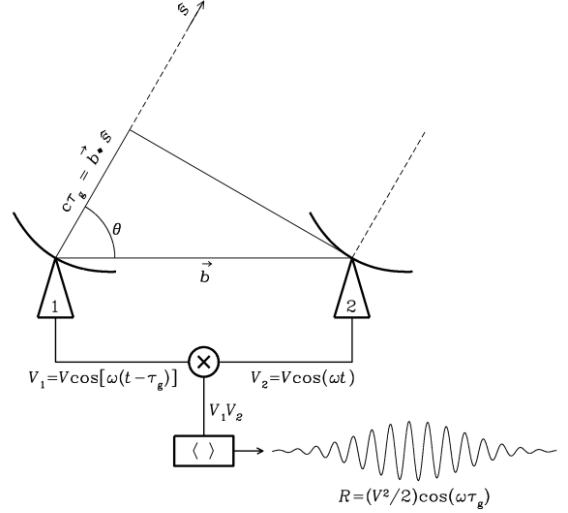


Figure 1: Two antennas with radio waves coming in parallel at an angle θ . The distance between the stations is the baseline vector, \mathbf{b} . $\hat{\mathbf{s}}$ is the unit vector in the direction of the source. The extra distance the light has to travel to reach station 1 can be written as $\mathbf{b} \cdot \hat{\mathbf{s}}$. To get the delay time, which we call τ_g , this distance has to be divided by c . [Condon and Ransom, 2010]

$$V_1 = V \cos(\omega(t - \tau_g)) \quad \text{and} \quad V_2 = V \cos(\omega t).$$

First, these two voltages are multiplied to get the combined signal, the response function.

$$V_1 V_2 = V^2 \cos(\omega t) \cos(\omega(t - \tau_g)).$$

To write the response as two separate terms instead of one, Simpson's formula ¹ is used, resulting in the following equation,

$$V_1 V_2 = \frac{V^2}{2} \cos(2\omega t - \omega\tau_g) + \cos(\omega\tau_g).$$

The voltages are averaged over a time interval larger than the time of a single full oscillation, so $T \gg \frac{2\pi}{\omega}$ [Wilson et al., 2009], which corresponds to a timescale of about 1 – 30 seconds. In that time interval the object does not shift significantly across the sky, so the delay time τ_g hardly changes. The other cosine term, $2\omega t - \omega\tau_g$ does change significantly, due to its dependence on t . The wave will move up and down in amplitude around zero, so the average goes to zero. Hence by averaging over time the high frequency and time variable component of the response will be removed. [Dr. John McKean, 2015].

¹ $\cos(x) + \cos(y) = 2 \cos(\frac{x+y}{2}) \cos(\frac{x-y}{2})$

with $x = 2\omega t - \omega\tau_g$ and $y = \omega\tau_g$.

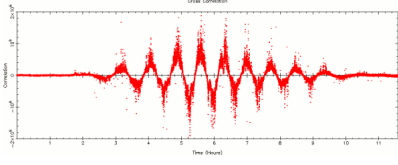


Figure 2: A fringe pattern, a pattern of evenly spaced alternating bright and dark bands, caused by the phase shift of the radio waves. [‘Fringe Dwellers’, 2015]

This results in,

$$\langle V_1 V_2 \rangle = \frac{V^2}{2} \cos(\omega \tau_g). \quad (1.2)$$

This quantity is defined as R , the response between two stations. Extending this approach to multiple stations, the signals from several different antennas are coupled together to produce an image. Instead of having one outcome for the response function, the function now has to be integrated over the whole solid angle. $\frac{V}{2}$ can be thought of as the intensity $I_\nu(\hat{s})$. At this point it becomes important that the intensity function can be described by two parts, an even and an odd part. When combining an odd response with a cosine, which is an even function, the final function will be odd as well, and will go to zero when integrating over the whole sky. For this reason the response function has to be split in two parts, one with a cosine which is sensitive to the even (symmetric) brightness, and one with a sine which is sensitive to the odd (asymmetric) part,

$$\begin{aligned} R_c &= \int I_\nu(\hat{s}) \cos(\omega \tau_g) d\Omega \\ R_s &= \int I_\nu(\hat{s}) \sin(\omega \tau_g) d\omega. \end{aligned}$$

The stations are still only sensitive to frequencies of $\nu = \frac{\omega}{2\pi}$. Using the identities $\tau_g = \mathbf{b} \cdot \hat{s}/c$ and $c = \lambda\nu$:

$$\begin{aligned} R_c &= \int I_\nu(\hat{s}) \cos(2\pi\nu \mathbf{b} \cdot \hat{s}/c) d\Omega \\ &= \int I_\nu(\hat{s}) \cos(2\pi \mathbf{b} \cdot \hat{s}/\lambda) d\Omega, \text{ and} \end{aligned}$$

Adding Interferometer Output

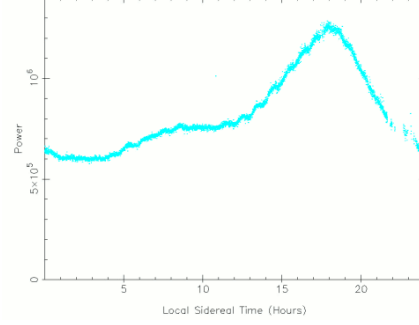


Figure 3: Output of an adding interferometer [‘Fringe Dwellers’, 2015]

$$\begin{aligned} R_s &= \int I_\nu(\hat{s}) \sin(2\pi\nu \mathbf{b} \cdot \hat{s}/c) d\Omega \\ &= \int I_\nu(\hat{s}) \sin(2\pi \mathbf{b} \cdot \hat{s}/\lambda) d\Omega. \end{aligned}$$

These two parts added together form the complex visibility, which is defined as $Z := R_c - iR_s$, so

$$Z = \int I_\nu(\hat{s}) \exp(-2\pi \mathbf{b} \cdot \hat{s}/\lambda) d\Omega. \quad (1.3)$$

By computing this formula for Z we assumed that the interferometer has the same sensitivity across the whole sky. In reality, the sensitivity of the beam response varies as a function of position, so the field of view is not constant over the whole sky. The sensitivity is maximum at the center, and tapers off towards the edges. To correct this another term has to be added to the visibility function Z , which corrects for the field of view [Burke and Graham-Smith, 2002]. We define A as the relative antenna response as a function of position, which will have a value of unity in the center of the point in the sky you are focusing on, and decrease in value as function of σ , the vector pointing from the center to the position of the source element. Including the relative antenna response, Z becomes

$$Z = \int A(\sigma) I_\nu(\hat{s}) \exp(-2\pi \mathbf{b} \cdot \hat{s}/\lambda) d\Omega. \quad (1.4)$$

To calculate the visibilities we multiplied the voltages of the two antennas. This used to be done differently. Old interferometers used a technique different than multiplying the voltages, called ‘adding interferometry’. This method only uses one receiver for both of the antennas, so the voltages V_1 and V_2 are added up. To get

the power measurement, the output needs to be squared, so $(V_1 + V_2)^2 = V_1^2 + V_2^2 + 2V_1V_2$. The only interesting term in interferometry is the cross term between the antennas V_1V_2 , but with this method this term is overshadowed by the signals of the separate antennas. Looking at the output of an adding interferometer, you see strongly the signal of one antenna, combined with a set of weak fringes, as shown in Figure 3. For this reason, multiplying interferometers are much more suitable than adding interferometers [‘Fringe Dwellers’, 2015, Wilson et al., 2009].

The measured visibilities are the inverse Fourier transform of the sky brightness distribution. The output of an interferometer gives a point in the uv -plane, and needs to be inverse-Fourier-transformed to get a good image of the sky. One combination of antennas gives only one point in the uv -plane, a set of many baselines, over a period of time, will give a better Fourier transform of the brightness distribution [Burke and Graham-Smith, 2002]. A method in which the rotation of the earth is used to make more measurement points as the source moves across the sky, sampling more of the uv -plane, is called aperture synthesis [Ker, 2010]. We want to measure the visibilities as accurately as possible, which means we need a large number of stations, as Section 1.4 will explain.

1.4 Independent closure phases

With more stations, the accuracy with which the interferometer measures the Fourier plane increases. This can be understood by noting that the number of measurements is equal to the number of baselines, which is given by $N_{\text{st}}(N_{\text{st}} - 1)/2$, with N_{st} the number of stations [Felli, M. and Spencer, R. E., 1989]. However, not all this data can be unique, there can also be redundant in-

formation in the data set. A good quantity to evaluate is the *closure phase*, which equals the sum of three phases around a closed triangle of baselines [Monnier, 2003]. This quantity is independent of phase shifts, because it is a closed path. No matter how much you change the position, in the end you always return at the starting point. The closure phase can be represented as follows:

$$\phi_{ijk} = \phi_{ij} + \phi_{jk} + \phi_{ki}. \quad (1.5)$$

Of interest is the number of *independent* closure phases, which is equivalent to holding one telescope fixed, and calculating all the possible closure phases between the others. The number of linearly independent combinations is given by [Thorsteinsson et al., 2004]

$$(N_{\text{st}} - 1)(N_{\text{st}} - 2) / 2. \quad (1.6)$$

To evaluate the quality of the information we can obtain from the telescopes, we can look at the ratio between the good observables and the total amount of baselines.

$$\frac{\text{independent closure phases}}{\text{baselines}} = \frac{N - 2}{N}. \quad (1.7)$$

For an interferometer with 3 stations, this ratio comes down to $(3 - 2) / 3 = 33\%$ of the information. In Table 1 the amount of information is shown for different numbers of telescopes.

From the increase in information for every station, we can conclude that it is essential to have enough stations in the interferometer, since each closed triangle between the baselines introduces a new independent closure phase, and increases the percentage of information.

Number of telescopes	Number of Fourier phases	Number of closing triangles	Number of independent closure phases	Percentage of phase information
3	3	1	1	33 %
7	21	35	15	71%
21	210	1330	190	90%
27	351	2925	325	93%
50	1225	19600	1176	96%

Table 1: Contained phase information for different numbers of telescopes. [Monnier, 2003]

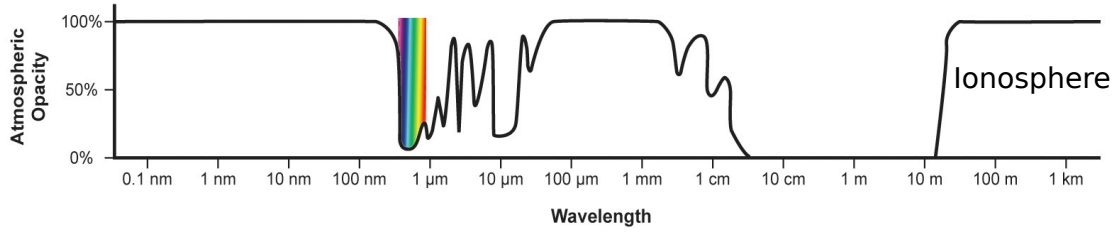


Figure 4: Atmospheric opacity as function of wavelength. At the right side of the image the cut-off from the ionosphere can be clearly seen. [Vector, 2015]

1.5 Ionosphere

When radio waves enter the atmosphere, they have to cross a region of partially ionized particles. This is called the ionosphere, and is created by the ultraviolet radiation of the Sun, which ionizes the particles entering the atmosphere. The ionosphere starts at a height above 100 km of the surface of the Earth, and stretches until about 600 km. Radio waves are influenced by the ionosphere, especially at low frequencies (< 1 GHz).

The free electrons in the ionosphere can collide with the photons of the radio waves, leading to reflection of the photons at low frequencies. At higher frequencies, the photons do not get reflected anymore, but the electromagnetic field of the free electrons and ions in the ionosphere still interacts with the radio waves, causing them to change phase. The influence of the ionosphere can be evaluated by looking at the refractive index of the ionosphere n ,

$$n = (1 - \nu_p^2/\nu^2)^{1/2}. \quad (1.8)$$

As can be seen in equation 1.8, the refractivity of the ionosphere increases as $1/\nu$, which means that radio waves with lower frequencies get more disturbed by the ionosphere, potentially making them difficult to detect² [Burke and Graham-Smith, 2002]. Figure 4 shows the atmospheric opacity as function of wavelength. At frequencies below about 10 MHz there can be total reflection, which is used as a communication method. This method is called 'DX communication', and uses the reflectivity at long wavelengths of the ionosphere to 'bounce' back signals, which enables you to send radio waves with information over a large distance. However, for radio astronomy it is not desirable when the data is being reflected back into space. Fortunately this reflection is a lot less severe at shorter wavelengths.

²In equation 1.8 the refractive index n is less than one, which implies a velocity $\frac{c}{n}$ of the wave greater than c . It is important to note that this corresponds to the *phase velocity* of the wave.

The resonance frequency ν_p is the value below which the atmosphere becomes opaque [Burke and Graham-Smith, 2002]. Its value is given by,

$$\nu_p^2 = \frac{Ne^2m^{-1}}{(2\pi\epsilon_0)^2}, \quad (1.9)$$

where N is the electron density, e is the electron charge, m is the electron mass and ϵ_0 is the vacuum permittivity. The electron density N depends on whether it is day or night, and on the solar activity [Spoelstra and Kelder, 1984]. Hence there is no sharp cut-off value for the resonance frequencies, they range from 4.5 MHz during the night to 11 MHz at daytime. Radio waves can only be detected accurately when their frequencies are well above this limit. LOFAR makes observations in two frequency ranges with two antennas, the Low Band Antenna (LBA) and High Band Antenna (HBA), optimized for 10 – 80 MHz and 120 – 240 MHz, respectively [LOFAR, 2015]. In the low frequency range, the effects of the ionosphere on the observations will be most noticeable, which means the data has to be corrected for them. Calibration of the data uses the measurement equation, which will be explained in Section 1.6.

1.6 Measurement equation

The propagation of radio waves through the different media, antennas and cables, up until the receiver can be described by an elegant equation, the *measurement equation*. Invented in 1996 by Johan Hamaker [Hamaker, J. P., 1996] and refined by Oleg Smirnov [Smirnov, 2011], the measurement equation is frequently used in calibration of data.

It is outside the scope of this thesis to take

into account that the antennas have in fact two polarization directions: x and y , which are perpendicular to each other. This means that when looking at the correlations between two antennas, there are 4 possible combinations: $p_x q_x$, $p_x q_y$, $p_y q_x$, and $p_y q_y$. Implementing this in the measurement equation would give 2×2 matrices in the matrices corresponding to the model and the data.

When not taking polarization into account, we can represent our data by a matrix with elements consisting of only complex visibility, instead of a 2×2 matrix. This matrix, \mathbf{V} contains the signals from sources in the sky as measured by the receiver. \mathbf{C} is our model matrix, which contains the ideal values of the data, before being corrupted by several effects. We can think of a gain matrix \mathbf{G} , which corresponds to the corruption factors that change \mathbf{C} into \mathbf{V} , such that,

$$\mathbf{V} = \mathbf{G} \mathbf{C}. \quad (1.10)$$

The gain matrix can be divided into other smaller gain matrices, each corresponding to a specific corruption effect,

$$\mathbf{V} = \mathbf{G}_n \mathbf{G}_{n-1} \dots \mathbf{G}_1 \mathbf{C}. \quad (1.11)$$

Looking at a single element of the \mathbf{V} matrix, V_{pq} , the signal travels along two different paths to the antennas p and q , which both have their own gain matrix³, \mathbf{G}_p and \mathbf{G}_q ,

$$V_{pq} = \mathbf{G}_{pn}^H (\dots \mathbf{G}_{p2}^H (\mathbf{G}_{p1}^H \mathbf{C} \mathbf{G}_{q1}) \mathbf{G}_{q2} \dots) \mathbf{G}_{qn}. \quad (1.12)$$

In this thesis we have taken the separate gain effects into one gain matrix,

$$\mathbf{V} = \mathbf{G}^H \mathbf{C} \mathbf{G}. \quad (1.13)$$

We are going to extend this approach to multiple directions, over which we will take a sum. The adapted version of the measurement equation is given by,

$$\mathbf{V} = \sum_d \mathbf{G}_d^H \mathbf{C}_d \mathbf{G}_d, \quad (1.14)$$

where d represents the different directions. In the next chapter we will explore this equation, and implement it in a calibration method. In Chapter 3 the simulations of direction dependent calibration will be presented. Possible future work will be discussed in Chapter 4 and the conclusions will be presented in Chapter 5.

³In equation 1.12, \mathbf{G}^H denotes the Hermitian transpose, which transposes the matrix and takes the complex conjugate.

2 Calibration

2.1 Introduction

Calibration is an essential part of radio interferometry. Uncalibrated data will show distortions and circles in the image, and will make it more difficult to see faint sources. Figure 5 shows the difference between a calibrated and an uncalibrated image. The current used calibration method 'StEFCal' (Statistically Efficient and Fast Calibration) is not direction dependent. However, the ionosphere is not homogeneous, so calibration should also vary as a function of position. In this paper we will explain the StEFCal algorithm and extend it to multiple directions.

Several effects corrupt the signal that comes from astronomical sources. Examples are the electrical gains, inexact compensation for cable lengths and the phase delay coming from the ionosphere. All these effects are specific to a station, and in the measurement equation they are represented by different matrices. For this thesis, all the effects are taken together in one gain matrix instead of several different ones, to make the calculations easier.

Each station has one gain factor, which has to be calculated from the correlations of that station with the other stations. The number of responses between one station and the others is much larger than the number of gains for a single station (which is one), meaning we are dealing with an over-determined system, which has more constraints than unknowns. This should always be the case, otherwise you could let your data correspond to any given model, and this will not allow you to make predictions about future data.

This phenomenon is known as overfitting. Before continuing with the calibration, the method of solving an over-determined system should be known, as is explained in an appendix, 7.1.

The gain factors for the stations can be calculated using a known model of the visibilities of the sky, determined from other surveys. The observed data set, the model and the gain factors can all be represented by matrices. The visibility matrix \mathbf{V} contains the correlation between all possible stations, where the element \mathbf{V}_{pq} gives the correlation between stations p and q . The model matrix \mathbf{C} contains the ideal value of those combinations of stations, and the gain matrix \mathbf{G} is a matrix with the correction factors between \mathbf{V} and \mathbf{C} . Both the rows and the columns of the model matrix have to be multiplied by the gain factors to get the measured data, this is achieved by putting a diagonal gain matrix in front of the model matrix, to multiply the rows, and one after the model matrix, to multiply the columns. In fact, this means we have a quadratic equation, which is difficult to solve for matrices. To find a solution the Alternating Direction Implicit method is used. We make an estimate for the first gain matrix, and use that to calculate the second, which we then take as the best estimate for the first matrix to calculate the next second gain matrix. Eventually this iterative process should make the first and second gain matrices approximately equivalent.

In Section 2.2 the process of non-directional calibration will be explained. In Section 2.3 we also take into account snapshots from multiple frequencies and in Section 2.4 we expand the calibration method to multiple directions.

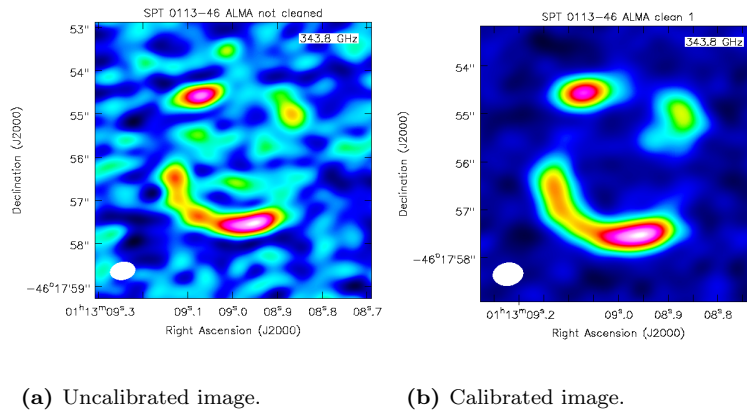


Figure 5: Two images of SPT0113-46 [Nobels, F. S. J., 2015].

2.2 Non-directional calibration

First, we will look at the calibration of the stations in one direction, by assuming that the ionosphere causes an equal phase shift in all directions. We have a certain data set, \mathbf{V} , where \mathbf{V}_{pq} contains the correlated data between stations p and q . We also have a specific model of the sky, \mathbf{C} , with \mathbf{C}_{pq} the ideal value that the correlated data should have, which is obtained from other surveys. Due to some properties of the station, the ionosphere through which light has to travel and some other factors, the data may deviate from the model, by a certain factor per station. \mathbf{G} is the gain matrix: a diagonal matrix with elements that change the model to make it look like the data we get out of the different stations of LOFAR. The first gain matrix multiplies each row with a certain number, and the second gain matrix multiplies each column with that number. The measurement equation can be visualized as shown in Figure 6.

$$\mathbf{V} = \mathbf{G}^H \mathbf{C} \mathbf{G}$$

Figure 6: Representation of the measurement equation, $\mathbf{V} = \mathbf{G}^H \mathbf{C} \mathbf{G}$.

We want our equations to satisfy $\mathbf{V} = \mathbf{G}^H \mathbf{C} \mathbf{G}$. The error in this equation is given by $\mathbf{V} - \mathbf{G}^H \mathbf{C} \mathbf{G}$, which means we want to minimize $\|\mathbf{V} - \mathbf{G}^H \mathbf{C} \mathbf{G}\|_F^2$. At the start of the algorithm, the first gain matrix is assumed to be known. We define the known gain matrix $\mathbf{H} = \mathbf{G}$. Our first estimate for \mathbf{H} is the identity matrix, with the same dimension as our data and model. Minimizing the error can be done by minimizing the Frobenius norm⁴, which is given by,

$$\ast = \left\| \mathbf{V} - \mathbf{H}^H \mathbf{C} \mathbf{G} \right\|_F^2$$

Figure 7: $\|\mathbf{V} - \mathbf{H}^H \mathbf{C} \mathbf{G}\|_F^2$,

⁴The Frobenius norm takes the squared norm of the absolute value of all the elements in the matrix, and sums over them. This way the negative and positive values will not cancel each other. Additional explanation can be found in Appendix 7.1

⁵The Matlab notation $:$ is used for 'all values for this index'.

where F denotes the Frobenius norm and \ast equals the Frobenius norm of the error. We want to find the \mathbf{G} for which this Frobenius norm has a minimal value. As the Frobenius norm is the sum of all elements, we can take the sum for every column separately⁵, and then sum over them, such that,

$$\ast = \sum_{k=1}^{\text{num. st.}} \left\| \mathbf{V}_{:k} - \left[\mathbf{H}^H \mathbf{C} \mathbf{G} \right]_{:k} \right\|_F^2$$

Figure 8: $\sum_{k=0}^N \|\mathbf{V}_{:k} - [\mathbf{H}^H \mathbf{C} \mathbf{G}]_{:k}\|_F^2$,

Since \mathbf{H}^H and \mathbf{C} are both known, they can be taken together in one matrix $\mathbf{A} := \mathbf{H}^H \mathbf{C}$.

$$\ast = \sum_{k=1}^{\text{num. st.}} \left\| \mathbf{V}_{:k} - \left[\mathbf{A} \mathbf{G} \right]_{:k} \right\|_F^2$$

Figure 9: $\sum_{k=0}^N \|\mathbf{V}_{:k} - [\mathbf{A} \mathbf{G}]_{:k}\|_F^2$.

Here, \mathbf{G} is a diagonal matrix, so the k^{th} column only has zero elements except for the element on the diagonal. This means that instead of making a diagonal matrix for \mathbf{G} , we can store the same information in a vector \mathbf{g} with length k . In other words, $\mathbf{g}_k = \mathbf{G}_{kk}$.

Now, if we look at $[\mathbf{A} \mathbf{G}]_{:k}$, we see that this is a matrix whose columns are the columns of \mathbf{A} multiplied by $\mathbf{G}_{kk} = \mathbf{g}_k$. So the k^{th} column of this matrix is given by $[\mathbf{A} \mathbf{G}]_{:k} = \mathbf{A}_{:k} \mathbf{g}_k$. So the equation becomes:

$$\ast = \sum_{k=1}^{\text{num. st.}} \left\| \mathbf{V}_{:k} - \mathbf{A}_{:k} \mathbf{g}_k \right\|_F^2$$

Figure 10: $\sum_{k=0}^N \|\mathbf{V}_{:k} - \mathbf{A}_{:k} \mathbf{g}_k\|_F^2$,

Each element of this sum is positive, since they are the square of the Frobenius norm. Hence

minimizing this sum is the same as minimizing each column separately. This is equal to finding the least squares solution to the problem $\mathbf{A}_{:k} \mathbf{g}_k = \mathbf{V}_{:k}$, which we already know how to solve from appendix 7.1, such that,

$$\begin{aligned}\mathbf{A}_{:k}^H \mathbf{A}_{:k} \mathbf{g}_k &= \mathbf{A}_{:k}^H \mathbf{V}_{:k} \\ \mathbf{g}_k &= (\mathbf{A}_{:k}^H \mathbf{A}_{:k})^{-1} \mathbf{A}_{:k}^H \mathbf{V}_{:k}\end{aligned}$$

In this case, since $\mathbf{A}_{:k}^H \mathbf{A}_{:k}$ and $\mathbf{A}_{:k}^H \mathbf{V}_{:k}$ are both scalars, we can simply calculate \mathbf{g}_k by dividing them,

$$\mathbf{g}_k = \frac{\mathbf{A}_{:k}^H \mathbf{V}_{:k}}{\mathbf{A}_{:k}^H \mathbf{A}_{:k}}$$

The obtained value for \mathbf{g}_k is placed in the vector \mathbf{g} , this is repeated for every k .

When every element of the vector \mathbf{g} is filled, it is used as the diagonal of the 'known' gain matrix to form the new \mathbf{H} . Each time this is done, the newly calculated gain vector gets closer to its true value, and the difference between the new and the old vectors gets smaller and smaller. For one directional calibration, convergence is proven by Stefano Salvini and Stefan Wijnholds [Salvini and Wijnholds, 2014]. The question is: at which point do we have to stop? To answer that we are going to use the relative improvement δ :

$$\delta = \frac{\|\mathbf{g}[i] - \mathbf{g}[i-1]\|_F^2}{\|\mathbf{g}[i]\|_F^2}. \quad (2.1)$$

For every even iteration we are going to check the relative improvement and when it is small enough this means that the old and new gain vector are almost identical, as they should be. At this point convergence is reached, the program stops and the latest version of \mathbf{g} will be kept.

2.3 Multiple data sets per station

In the previous section we only used one snapshot to calculate the gains, which is not really reliable. The more snapshots are used, the more accurate the gain matrices become. The signal to noise ratio scales with the square root of the number of data sets, $S/N \propto \sqrt{N}$.

So what happens when we take into account that each station has multiple data sets? We could, of

course, see the different data sets as a new dimension of the matrices, but this will make things unnecessarily complicated, especially when we go on to directional-calibration, since there will also be a new dimension of the direction.

Instead of creating an extra dimension for the frequencies, we will order the different data sets below each other, so the matrix corresponding to the data \mathbf{V} will become tall. Each data set could correspond to a different frequency band, for which the model \mathbf{C} is different as well. We do the same thing for \mathbf{C} , and put every different model below each other in a tall matrix. The dimension of these matrices depends on the number of measurements made, M , and the number of stations, N_{st} , and is given by $(N_{\text{st}} \cdot M) \times M$.

The left gain matrix will then become an identity matrix with dimension the same as the length of \mathbf{V} and \mathbf{C} , so this matrix will be a lot larger than before. The right gain matrix will still be the original size. Just like before, the only information needed about the gain matrices is in the form of a vector with dimension N_{st} . To make the big gain matrix, these elements are put on the diagonal and repeated M times. We can do this because the gain matrix is the same for every data set, since it only depends on variables like the ionosphere, the error in the distance between the stations and their sensitivity, which are constant in a short time range (of about 1 second).

The equation $\mathbf{V} = \mathbf{G}^H \mathbf{C} \mathbf{G}$ now looks like,

$$\mathbf{V} = \mathbf{G}^H \mathbf{C} \mathbf{G}$$

and after introducing \mathbf{A} ,

$$\mathbf{V} = \mathbf{A} \mathbf{G}$$

Due to the rather large size of \mathbf{G}^H , converting the vector \mathbf{g} into a diagonal matrix to carry out the multiplication with \mathbf{C} would cost a lot of computing power. In this case all the zeros in the matrix would have to be multiplied with the elements of \mathbf{C} as well, which is time consuming and unnecessary. By using a smart method of calculating \mathbf{A} we can make the calculation run a lot faster. The operation carried out by multiplying \mathbf{C} with the left diagonal gain matrix is that each row gets multiplied with a certain number. This calculation can also be done using a different method, by multiplying element wise the gain vector \mathbf{g} with the matrix \mathbf{C} . The gain vector first needs to have the same length as \mathbf{C} , this is achieved by repeating the elements M times, such that,

$$\mathbf{A} = \mathbf{g} \odot \mathbf{C}$$

where \odot denotes the Hadamard (element-wise) product.

However, we can even go a step further by taking into account that for every step we only need to know one column of \mathbf{A} . We never need to use the whole matrix, so it is not necessary to compute it and we can suffice with,

$$\mathbf{A}_{:k} = \mathbf{g} \odot \mathbf{C}_{:k}.$$

If we only compute one column of \mathbf{A} at a time, this will save memory, and also order the elements we need in a vector, making them quick to access.

After computing \mathbf{A} in an efficient manner, the least squares solution can be found for \mathbf{g} , using the same method as in paragraph 2.2. This is demonstrated in the first script in Appendix 7.2. In the following Section we will extend this approach to 3 dimensions.

2.4 Directional calibration

The method presented in Section 2.2 is valid when you assume that the gains are the same in every direction of the sky. However, in reality this is not true, the ionosphere is different in every direction. This means we have to extend our approach to 3 dimensions, similar to the method derived in [Tasse, 2014]. The first two dimensions are the different stations, just as before, and the third dimension is the direction. We divide the sky into different sections, and per section we have a different model. Our data is still a 2D matrix, but our model and gains become 3D tensors,

since we added the extra dimension of the direction. For the clarity of this explanation the matrices are all represented as square matrices, but in reality the different frequency measurements are put beneath each other to produce tall matrices just like in Section 2.3. This means we can visualize the measurement equation as follows:

$$\left\| \begin{array}{c} \text{Data} \\ \mathbf{V} \end{array} - \sum_{d=1}^D \left[\begin{array}{c} \text{Gain}^H \\ \mathbf{G}^H \end{array} \begin{array}{c} \text{Model} \\ \mathbf{C} \end{array} \begin{array}{c} \text{Gain} \\ \mathbf{G} \end{array} \right] \right\|_F^2 = *$$

in which we want to minimize $*$. In this 3 dimensional variant of StEFCal the gain matrices have turned into diagonal tensors, where we still use the identity tensor as our initial estimate of \mathbf{G}^H .

Again, we define \mathbf{A} as $\mathbf{G}^H \mathbf{C}$,

$$\mathbf{A} = \begin{array}{c} \text{Gain}^H \\ \mathbf{G}^H \end{array} \begin{array}{c} \text{Model} \\ \mathbf{C} \end{array} = \begin{array}{c} 1 \times \\ 2 \times \\ 3 \times \\ \vdots \end{array} \begin{array}{c} \text{Model} \\ \mathbf{C} \end{array} \begin{array}{c} \text{Gain} \\ \mathbf{G} \end{array}$$

which means every row of \mathbf{C} gets multiplied by the corresponding value on the diagonal of \mathbf{G}^H . This gives us

$$\left\| \begin{array}{c} \text{Data} \\ \mathbf{V} \end{array} - \sum_{d=1}^D \left[\begin{array}{c} \text{Model} \\ \mathbf{A} \end{array} \begin{array}{c} \text{Gain} \\ \mathbf{G} \end{array} \right] \right\|_F^2 = *$$

Now, we will evaluate each column separately. In the previous example this resulted in a vector for \mathbf{A} , here it will reduce to a 2D matrix, which can be represented by

$$\sum_{k=1}^{\text{num. st.}} \left\| \begin{array}{c} \text{Data} \\ \mathbf{V}_{:k} \end{array} - \sum_{d=1}^D \left[\begin{array}{c} \text{Model} \\ \mathbf{A}_{:k} \end{array} \begin{array}{c} \text{Gain} \\ \mathbf{G}_{kk} \end{array} \right] \right\|_F^2 = *$$

Since we only need the columns of \mathbf{A} separately, we will again not compute the whole 3D matrix

but only use one plane of \mathbf{A} at a time. For easier notation, we define:

$$\mathbf{A}_{:,k} = \mathbf{A}_k = \text{[green vertical bar]}$$

How do we compute the matrix \mathbf{A}_k , in the most efficient way? Let us first look at a slice in one direction of the tensors \mathbf{G}^H and \mathbf{C} ,

$$\text{[green vertical bar]} = \text{[green diagonal line]} \odot \text{[green vertical bar]}$$

$\mathbf{A} \quad \mathbf{G}^H \quad \mathbf{C}$

Expanding this to D directions, we get

$$\text{[stack of green vertical bars]} = \text{[green diagonal plane]} \odot \text{[stack of green vertical bars]}$$

$\mathbf{A} \quad \mathbf{G}^H \quad \mathbf{C}$

We will sum this over every k . Putting it back in our original equation, we have

$$\sum_{k=1}^{\text{num. st.}} \left\| \text{[green vertical bar]} - \sum_{d=1}^D \left[\text{[green diagonal plane]} \right] \right\|_F^2 = *$$

$\mathbf{V}_{:,k} \quad \mathbf{A}_k \quad \mathbf{G}_{kk,:}$

When we write this down in two dimensions, we get the following equation.

$$\sum_{k=1}^{\text{num. st.}} \left\| \text{[green vertical bar]} - \sum_{d=1}^D \left[\text{[green rectangle]} \right] \right\|_F^2 = *$$

$\mathbf{V}_{:,k} \quad \mathbf{A}_k \quad \mathbf{G}_{kk,:}$

Where $\mathbf{G}_{k k,:}$ is a vector with the directional elements of the gain matrix, as seen below. Although \mathbf{G} is a 3D tensor, all its information is stored in the diagonal, since the other elements are all equal to zero. This means that we can

make a 2D matrix which represents the 3D tensor \mathbf{G} , and contains all the diagonal elements. This is a $D \times N_{\text{st}}$ matrix. Its columns will be filled with the vector $\mathbf{G}_{k k,:}$. To make the notation easier, we define $\mathbf{G}_{k k,:} := \mathbf{g}_k$

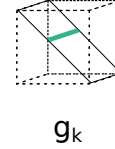


Figure 11: Representation of $\mathbf{G}_{k k,:} = \mathbf{g}_k$.

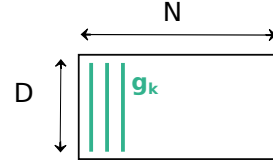


Figure 12: The matrix \mathbf{G} .

For every k , we have to find the \mathbf{g}_k that minimizes $*$. Finding the minimum of $\|\mathbf{V}_{:,k} - \mathbf{A}_k \mathbf{g}_k\|_F^2$ is the same as finding the least squares solution of $\mathbf{A}_k \mathbf{g}_k = \mathbf{V}_{:,k}$. This means we have to solve this equation

$$\text{[green rectangle]} \text{ [green vertical bar]} = \text{[green vertical bar]}$$

$\mathbf{A}_k \quad \mathbf{g}_k \quad \mathbf{V}_{:,k}$

with the least squares method, as explained in Appendix 7.1. In this way we calculate \mathbf{g}_k for every k , and we fill the columns of the matrix \mathbf{G} with the solutions. We repeat the process i times, using the conjugate of the last calculated \mathbf{G} matrix as the left matrix \mathbf{G}^H , to calculate \mathbf{A}_k .

When the gain matrices converge, we use the final result to correct the data set \mathbf{V} . This gives us the output we want, the set of LOFAR data corrected for most of the influence of the ionosphere.

In this chapter we have seen the theory behind the StEFCal algorithm, first for the non-directional variant and afterwards expanded to multiple directions. The implementation of this method can be seen in the second script in Ap-

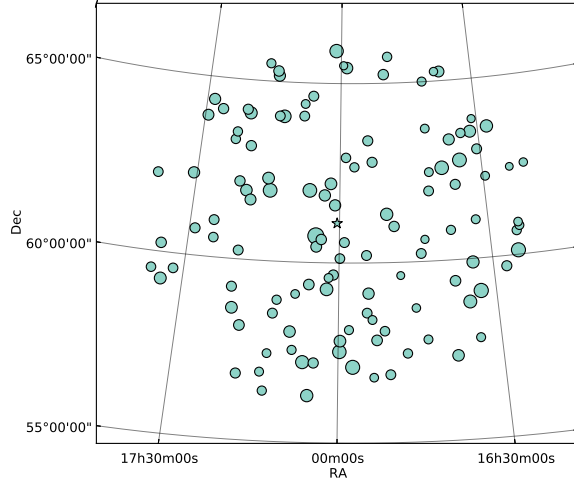
pendix 7.2. Using this script we evaluated real data from LOFAR, to investigate the efficiency of the algorithm. The findings will be discussed in the next chapter.

3 Experimental results

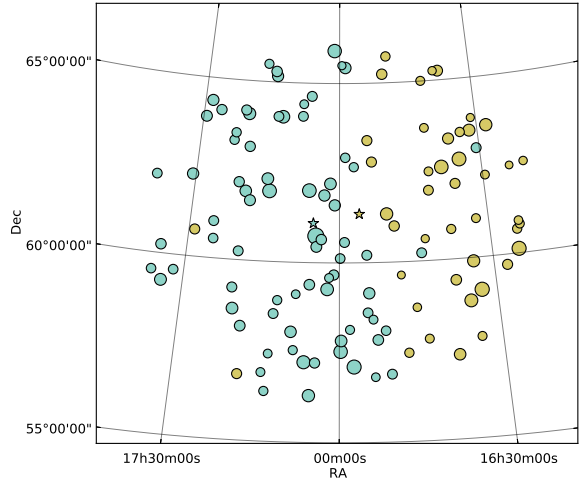
3.1 Variable number of directions

As a test observation, we have used one time slot of a sub band of a MSSS verification field [MSSS, 2011]. The source model we calibrate for comes from the calibration catalog for MSSS, the data was kindly provided by George Heald [Heald and LOFAR Collaboration, 2014]. The prediction of these sources was performed using the LOFAR software package Black Board Self-calibration (BBS).

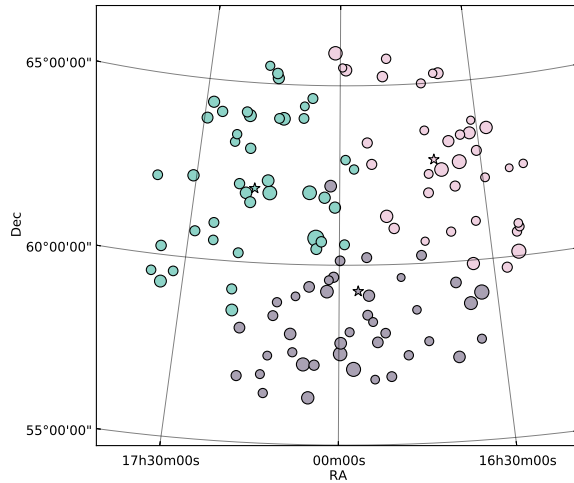
How exactly did we divide the sky into different directions? For the grouping of the sources the SAGECAL clustering algorithm [SAGECAL, 2015] as implemented in 'LSMTool' was used, which is an algorithm that groups sources into a specified number of clusters. It divides sources according to their position in the sky, but also takes into account that the flux density should be distributed evenly amongst the different sections. To investigate which number of directions is optimal, plots have been made with the number of directions varying from 1 to 5.



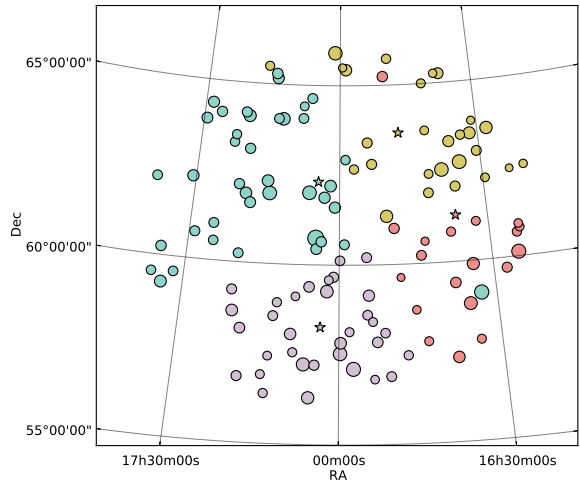
(a) 1 direction.



(b) 2 directions.

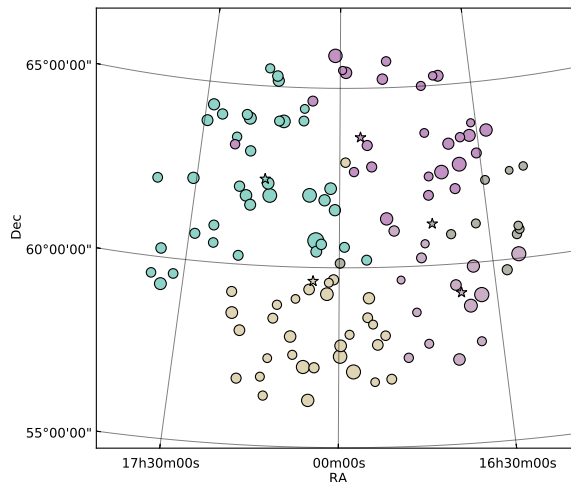


(c) 3 directions.



(d) 4 directions.

Figure 13: Grouping of the sources, for 1 to 5 directions.



(e) 5 directions

Figure 13 shows the division of sources in the sky made by the clustering algorithm for several different numbers of directions. The different colors correspond to different directions.

3.2 Convergence plots

After several iterations of the calibration procedure for the MSSS data set, the two gain matrices should be almost identical. The relative improvement of the two, as defined in equation 2.1, is used to measure their convergence. Convergence plots show how δ improves as a function of the number of iterations. The lower the value of δ , the more the calculated gain matrices resemble each other.

3.2.1 Dependence on number of directions

The different number of directions as seen in Section 3.1 can be compared to each other by looking at their convergence plots. In Figure 14 it can be seen that for one direction, the gains converge quickly to $\delta \approx 10^{-17}$ after approximately 50 iterations. For multiple directions, the gains converge a lot slower, after approximately 1000 iterations. Remarkable about the plots is that for direction dependent calibration, the convergence does not seem to depend strongly on the number of directions. The plots are all fairly similar. This may imply that for a higher amount of directions than 5, calibration will not be extremely more difficult, suggesting that directional calibration

with StEFCal is a promising method.

At the interval of $i \approx 100 - 300$, the plots of 4 and 5 directions behave strangely. They deviate strongly from a straight line for some period, and afterwards they behave regularly again. We are not sure what can explain this behavior, it could be because of ambiguity. The algorithm may have trouble finding the right gains solution, if there are several solutions that are correct. At some point it seems the algorithm has found the right values, and from then on the convergence speeds up. Further study is required to investigate this hypothesis.

The slopes of the convergence plots were calculated with Python in log-log scale. The number of iterations ranges from 0 to 60, because in that interval the plots behave approximately as a straight line. The results are shown in Table 2. Since the slopes are calculated in log-log scale, we can conclude from the values of the slopes that for non-directional StEFCal, the relative improvement scales with the number of iterations as $\delta \propto i^{-12}$. For direction dependent StEFCal, they scale approximately as $\delta \propto i^{-2}$.

In the following plots, other parameters will be varied and the number of directions will stay the same. We chose to make the plots with 5 directions, this is an arbitrary choice, since the convergence does not depend strongly on the number of directions.

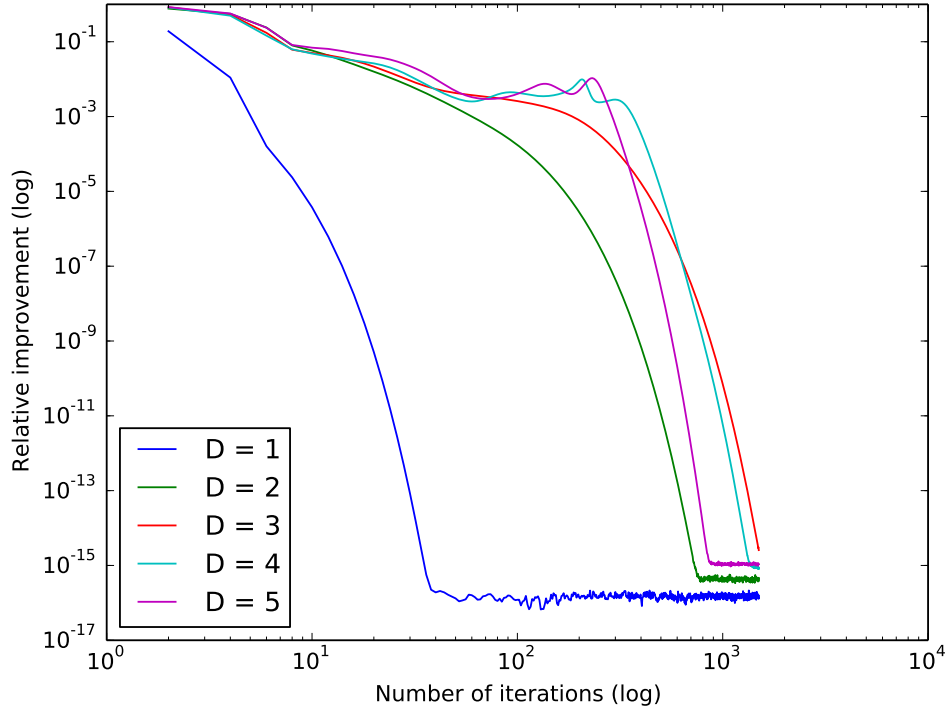


Figure 14: Convergence plot for different number of directions.

Number of Directions	Slope
1	-12.9
2	-2.1
3	-1.7
4	-1.8
5	-1.7

Table 2: Slope of the log-log convergence plots for multiple different numbers of directions, calculated with Python using a fitting function. As can be seen, the slope for 1 directions is a lot steeper than the ones for multiple directions.

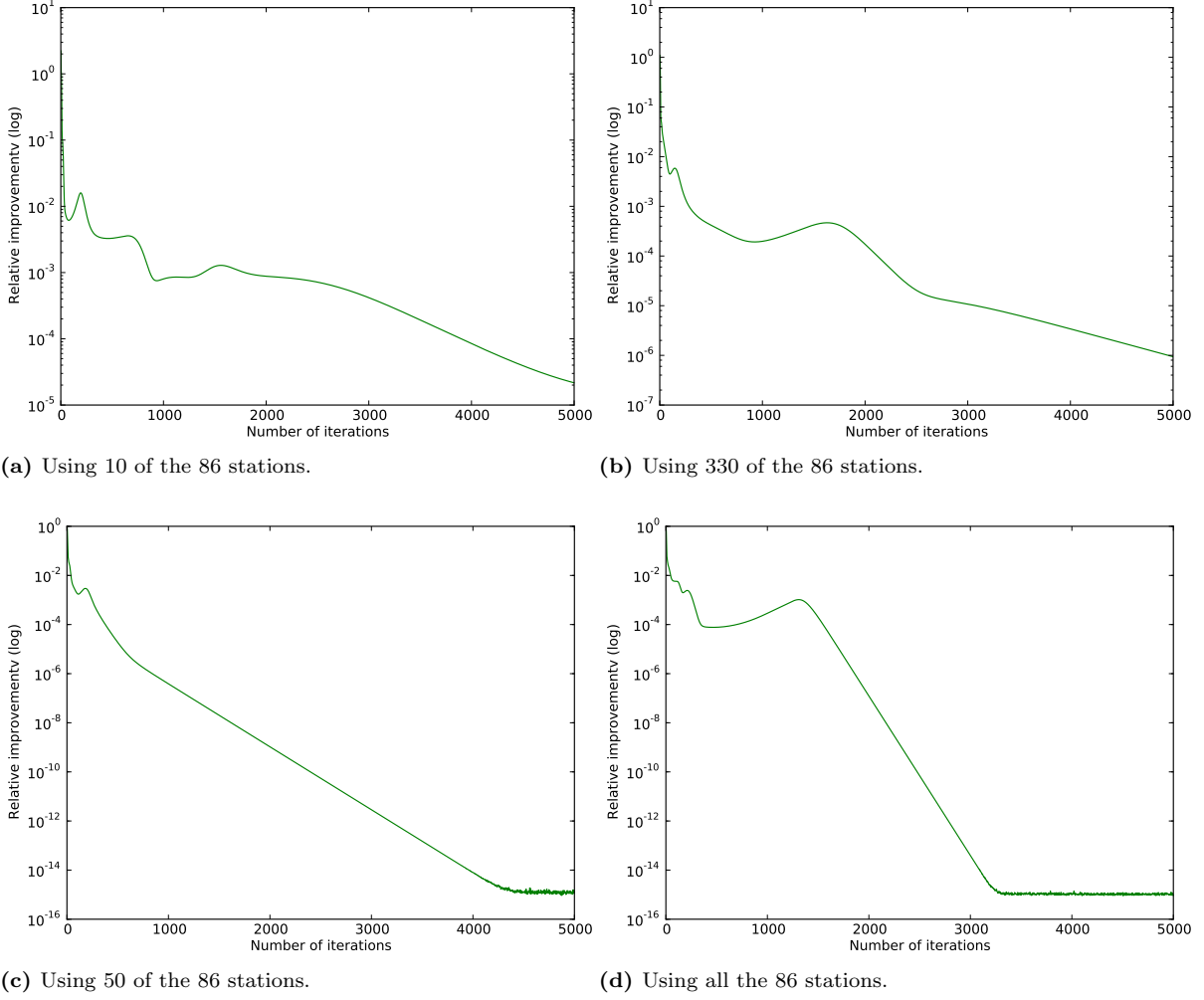


Figure 15: Convergence plots for varying number of stations.

3.2.2 Dependence on number of stations

In figure 15 we show a subset of the data, that does not contain every station. We can see that the process improves when the number of stations increases, as the point of convergence, $\delta \approx 10^{-15}$, is reached sooner. There is a direct relation between the relative improvement and the number of iterations needed to reach convergence on a log-linear scale, which implies a relation between δ and the number iterations of the form $\delta \propto 10^{-i}$. The first 1500 iterations do deviate from this, however.

3.2.3 Averaging the frequencies

Another interesting property to investigate is what happens when we average the frequencies. Instead of putting all the frequencies beneath each other in a tall matrix, now we are going

to make one matrix of $N_{st} \times N_{st}$ which contains all the visibilities averaged over the different frequencies. In Figure 16 both the method used before of making a tall matrix and the averaged frequencies method are shown. The results of the averaged frequencies are not as good as the previous used ones. They do converge, but they need a higher amount of iterations, about 9000.

3.2.4 Without the relaxation process

The crucial idea of StEFCal is to implement a relaxation process, which replaces the gain solution of each even iteration by the average of the current gain solution and the gain solution of the previous odd iteration. This process helps enormously to reach convergence. The result of omitting this relaxation process is demonstrated in Figure 17, and it can be seen that the gains do not converge at all anymore.

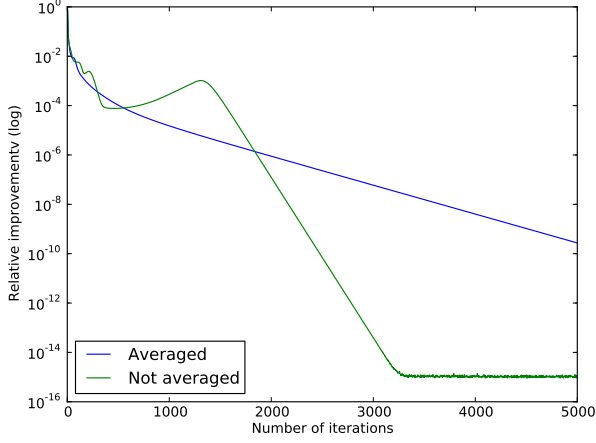


Figure 16: Convergence plot with the visibilities averaged over the frequencies.

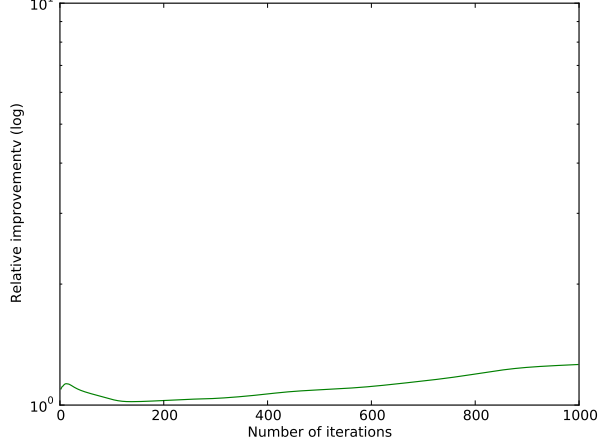


Figure 17: Convergence plot without the relaxation process.

3.2.5 Individual gain elements

Instead of taking the relative improvement of the total matrix, we can also evaluate one element at a time. This could be interesting because it enables us to obtain a better idea of how the individual components of the matrix behave. If some elements possess a larger error, this could propagate into the final gain matrix as well. The first stations of the matrix correspond to the core stations, they are located close to each other, so their baselines are short. The last stations correspond to the distant ones, with longer baselines. It would be interesting to investigate whether they demonstrate different behavior.

Figure 18 shows the convergence plots of the distant stations. All the different directions seem to behave similarly, the plots show no remarkable distinctions.

In Figure 19 both the 1st and the 86th elements are plotted for every direction. These plots demonstrate a significant difference between the core stations and the remote stations. The distant stations, which are located far away from each other, converge a lot better than the core stations. The core stations do converge, eventually, but because they need such a high amount of iterations they slow down the convergence of the total gain matrix.

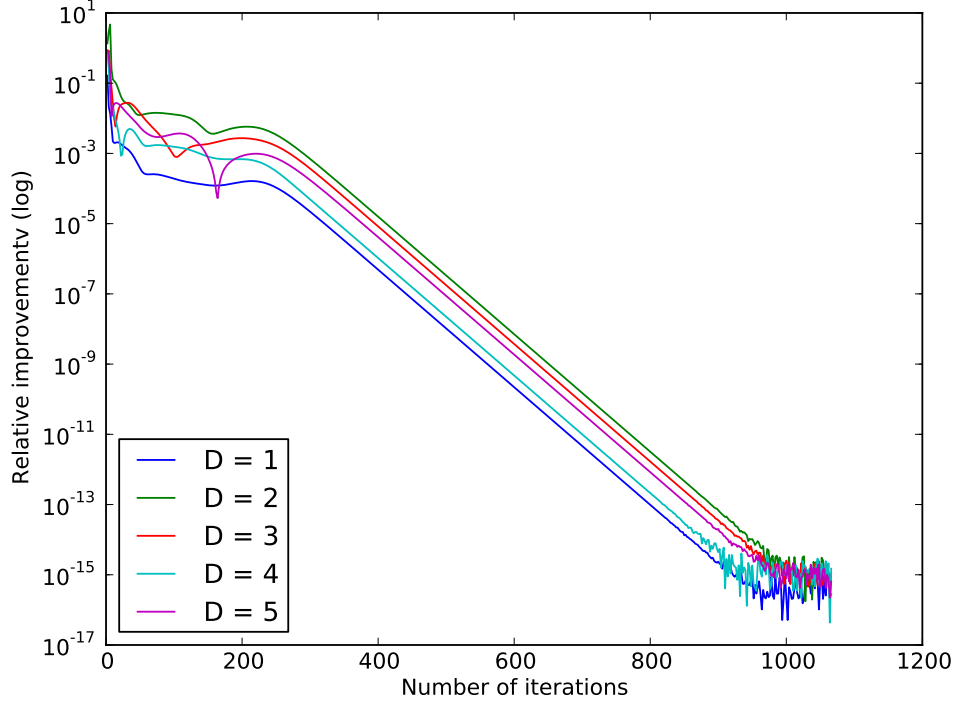


Figure 18: Convergence plots of only the 86th element of the gain matrix, plotted for every direction.

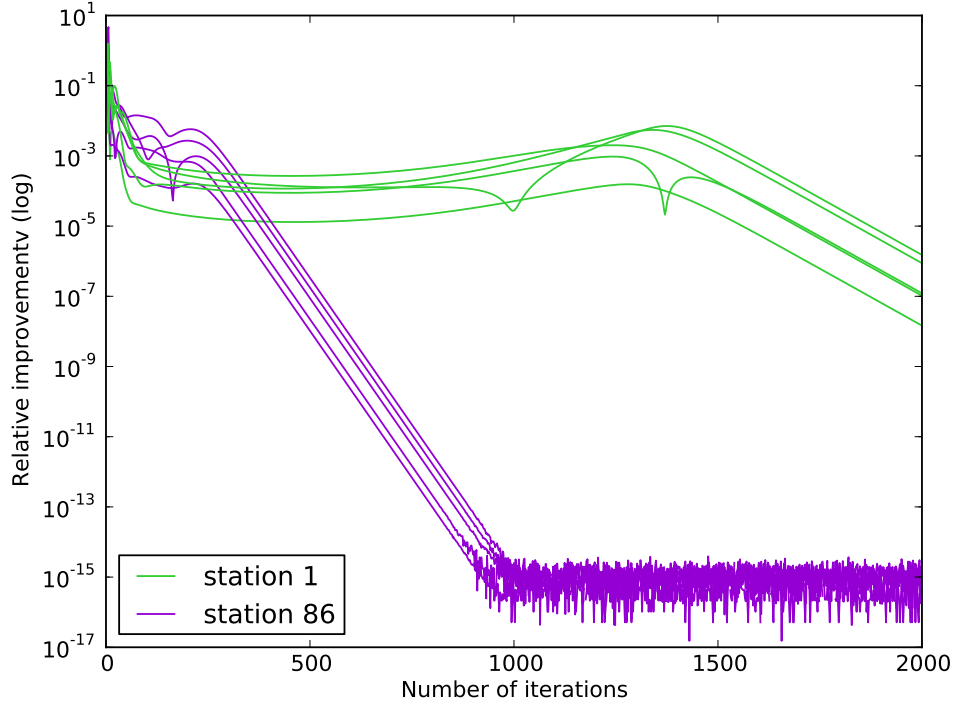


Figure 19: Convergence plots for both the 1st and the 86th elements, plotted for every direction. A significant difference between the 1st and the 86th can be seen.

4 Discussion and future research

When calibrating the data, the measurement equation is solved for the total gain matrices. This way we find the total distortion factor of the data, which consists of several effects taken together. It would be useful to make a distinction between those, since some effects follow an identifiable relation, for instance the effects of the ionosphere scale with $1/\nu^2$, and the delay as an effect of the position of the station scales with $1/\nu$. Also, some of the effects are not direction dependent. If we would break the gain matrices up into several different matrices that each contain the gains of a certain effect, we could evaluate them separately and make it easier to see small deviations.

The relaxation method applied in StEFCal proved to be efficient. It would be interesting to see if other relaxation methods could obtain the same effect, or maybe even a better one. In the directional variant, convergence was not very good, there might be another relaxation scheme that could speed this process up. Also, it could be evaluated whether it is necessary to keep applying the relaxation method, or whether at some point it is not needed anymore. This would save some computing time.

On the topic of time efficiency a lot of further progress can be made as well, to make the algorithm as quick as possible. In the first scripts the Moore-Penrose pseudo inverse function of python was used to calculate the gain vector. On suggestion of Stefano Salvini this was replaced by a function which immediately returns the least-squares solution to a linear matrix equation, without computing the matrix explicitly. This alteration of the program made it run significantly faster and made the difference between waiting the whole day on a result, to receiving it in a few minutes. Hence it would be a good investment to look critically at some of the other functions and methods used, and make the program run as fast as computationally possible.

Only data from one time slot was calibrated, with multiple directions and multiple frequencies. Additional information could be obtained when we look at multiple time slots, to improve the signal-to-noise ratio.

More information about the gain matrix can be obtained by looking at the separate elements. In this thesis the 1st and the 86th element have been compared and the results show that the gain elements of the 86th station converge significantly faster than those of the 1st station. Future research could find out more about the behavior of the other stations, and by doing so get a better understanding of the total gain matrix.

5 Conclusion

In this thesis, a direction dependent extension was derived for StEFCal, that calculates the gain matrices for a specific number of stations, frequencies and directions. The corrected visibilities can be found by multiplying these gain matrices with the model. In this research project both the speed and the accuracy of the algorithm have been evaluated by producing convergence plots, in which the relative improvement δ is plotted as a function of the number of iterations.

From the plots we find that the relative improvement reaches a point of convergence of about 10^{-15} for 5 directions, and 10^{-17} for one direction. However, for directional calibration it does take a high number of iterations to achieve this value. It has been found that for directional calibration, the number of directions you choose does not have a big influence on the results. The graphs give a line that is straight by approximation, after the first 1000 iterations, when $\log(\delta)$ is plotted against i (the number of iterations). This indicates a relation between δ and i of the form $\delta \propto 10^{-i}$.

An experiment was done with averaging the visibilities over the frequencies, instead of calculating the gains with every single frequency matrix. This turned out to take a lot more iterations to reach convergence than the previous approach.

When the StEFCal relaxation method is left out of the algorithm, the gains do not converge anymore. From this we can conclude that the StEFCal approach is indeed very effective.

Comparisons between the core stations and the distant stations give as result that the gains of the distant stations converge much faster than those of the core stations.

References

- [Burke and Graham-Smith, 2002] Burke, B. F. and Graham-Smith, F. (2002). *An Introduction to Radio Astronomy: Second Edition*.
- [Condon and Ransom, 2010] Condon, J. J. and Ransom, S. M. (2010). Essential Radio Astronomy. www.cv.nrao.edu/course/ast534.
- [Dr. John McKean, 2015] Dr. John McKean (2015).
- [Felli, M. and Spencer, R. E., 1989] Felli, M. and Spencer, R. E. (1989). *Very Long Baseline Interferometry*.
- ['Fringe Dwellers', 2015] 'Fringe Dwellers' (2015). Fringe Dwellers. <http://fringes.org/>.
- [Hamaker, J. P., 1996] Hamaker, J. P. (1996).
- [Heald and LOFAR Collaboration, 2014] Heald, G. and LOFAR Collaboration (2014). The LOFAR Multifrequency Snapshot Sky Survey (MSSS): Status and Results. In *American Astronomical Society Meeting Abstracts #223*, volume 223 of *American Astronomical Society Meeting Abstracts*.
- [Ker, 2010] Ker, L. M. (2010). An Introduction to Radio Interferometry and The Measurement Equation Formalism, Pedagogical Seminar.
- [LOFAR, 2015] LOFAR (2015). Technical Descriptions LOFAR.
- [Monnier, 2003] Monnier, J. D. (2003). Astrophysics with Closure Phases. In Perrin, G. and Malbet, F., editors, *EAS Publications Series*, volume 6 of *EAS Publications Series*, page 213.
- [MSSS, 2011] MSSS (2011). Multifrequency Snapshot Sky Survey. <https://www.astron.nl/radio-observatory/lofar-msss/lofar-msss>.
- [Nobels, F. S. J., 2015] Nobels, F. S. J. (2015). <http://www.astro.rug.nl/~nobels/index.php>.
- [SAGECAL, 2015] SAGECAL (2015). SAGECAL clustering algorithm.
- [Salvini and Wijnholds, 2014] Salvini, S. and Wijnholds, S. J. (2014). Fast gain calibration in radio astronomy using alternating direction implicit methods: Analysis and applications. 571.
- [Smirnov, 2011] Smirnov, O. M. (2011). Revisiting the radio interferometer measurement equation. I. A full-sky Jones formalism. 527.
- [Spoelstra and Kelder, 1984] Spoelstra, T. A. T. and Kelder, H. (1984). Effects produced by the ionosphere on radio interferometry. *Radio Science*, 19:779–788.
- [Tasse, 2014] Tasse, C. (2014). Applying Wirtinger derivatives to the radio interferometry calibration problem. *ArXiv e-prints*.
- [Thorsteinsson et al., 2004] Thorsteinsson, H., Buscher, D. F., and Young, J. S. (2004). The bispectrum in model-independent imaging. In Traub, W. A., editor, *New Frontiers in Stellar Interferometry*, volume 5491 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 1574.
- [Vector, 2015] Vector (2015). Vector. <http://www.vectorhq.com/vector/atmospheric-electromagnetic-opacity-113363>.
- [Wilson et al., 2009] Wilson, T. L., Rohlfs, K., and Hüttemeister, S. (2009). *Tools of Radio Astronomy*. Springer Berlin Heidelberg, Berlin, Heidelberg .:

6 Acknowledgements

I would like to thank my supervisor Tammo Jan Dijkema for his enthusiasm, his explanations of the subject, for finding time in his busy schedule to help me and for his dedication to the project. My other supervisor John Mckean has also helped me a lot, by explaining the astrophysics behind the algorithm with clarity and inspiring enthusiasm. I also want to thank my fellow astronomy students and friends for motivating me, making my workplace feel like a home, and providing me with licorice when needed. In particular I want to thank Eifion Prinsen for his feedback and Folkert Nobels for his never ending help with installing linux on my laptop, with L^AT_EX, for checking my thesis and much more. I am also really happy that ASTRON gave me the chance to do this project and provided me with a work spot. Thank you Wouter, Rik, Leandro, Matthias and Adriaan for driving me to ASTRON, taking me on lunch walks in the forest and making me see that work-life can be fun. In particular I want to thank Wouter Klijn for his valuable feedback. Stefano Salvini and Stefan Wijnholds, it was really nice to meet the authors of the paper I based my work on, thank you for inspiring me and making me laugh! Last but not least, I would like to thank Jolien Diekema and the rest of the 'SLEF' committee of 'T.F.V. Professor Francken' for giving me the opportunity to do this project.



7 Appendix

7.1 Solving over-determined systems

Imagine a set of equations with equal dimensions of the form $\mathbf{Ax} = \mathbf{b}$, as in the following example.

$$\begin{pmatrix} 5 & 2 \\ 4 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 10 \end{pmatrix}$$

This can be solved by multiplying each side of the equation with the inverse of \mathbf{A} .

The solution is given by:

$$\mathbf{x} = \mathbf{A}^{-1} \mathbf{b} \quad (7.1)$$

$$\mathbf{A}^{-1} = \begin{pmatrix} 1 & -1 \\ -2 & 2\frac{1}{2} \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -2 & 2\frac{1}{2} \end{pmatrix} \begin{pmatrix} 4 \\ 10 \end{pmatrix} = \begin{pmatrix} -6 \\ 17 \end{pmatrix}$$

But what happens when you have more equations than unknowns? For example, consider the problem

$$\begin{pmatrix} 3 \\ 4 \end{pmatrix} x = \begin{pmatrix} 4 \\ 10 \end{pmatrix}$$

In this case we won't be able to come up with one \mathbf{x} that satisfies all the equations. There will always be an error, defined as:

$$\mathbf{r} := \mathbf{b} - \mathbf{Ax} \quad (7.2)$$

which needs to be minimized to give us the best result. Minimizing a vector has to be done in a norm, which translates a vector to a real number. For this we use the Frobenius norm, meaning we take the squared norm of the absolute value of all the elements of the vector, sum them and then take the square-root. This way the negative and positive values won't cancel each other. The Frobenius norm is a scalar, and is defined as:

$$\|\mathbf{A}\|_F^2 = \sum_i \sum_j |a_{ij}|^2 \quad (7.3)$$

The error \mathbf{r} is, as given by equation 7.2,

$$\mathbf{r} = \begin{pmatrix} 4 \\ 10 \end{pmatrix} - \begin{pmatrix} 3 \\ 4 \end{pmatrix} x = \begin{pmatrix} 4 - 3x \\ 10 - 4x \end{pmatrix}$$

To get the optimal value for x , we want to minimize the Frobenius norm of the error.

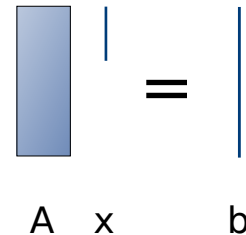
$$\|\mathbf{r}\|_F^2 = (4 - 3x)^2 + (10 - 4x)^2$$

This can be done by differentiating the Frobenius norm and setting its derivative equal to zero. The outcome for x is the value which gives a minimum error.

$$\begin{aligned} 0 &= \frac{\partial}{\partial x} \|\mathbf{r}\|_F^2 \\ 0 &= -6(4 - 3x) - 8(10 - 4x) \\ 0 &= -104 + 50x \\ x &= \frac{52}{25} \end{aligned} \quad (7.4)$$

When you have more parameters, however, computing the high-dimensional derivative becomes unpractical. Fortunately there is a better and more generic method, which will be explained here.

We have the set of equations $\mathbf{Ax} = \mathbf{b}$, with $\dim(\mathbf{b}) > \dim(\mathbf{x})$.



$$\mathbf{A} \mathbf{x} = \mathbf{b}$$

We want \mathbf{A} , \mathbf{x} and \mathbf{b} to have the same length, so we multiply both sides with the Hermitian

transpose of \mathbf{A} .

$$\begin{array}{ccc} \begin{array}{|c|} \hline \\ \hline \end{array} & \begin{array}{|c|} \hline \\ \hline \end{array} & \begin{array}{|c|} \hline \\ \hline \end{array} \\ \mathbf{A}^H & \mathbf{A} & \mathbf{x} \end{array} = \begin{array}{ccc} \begin{array}{|c|} \hline \\ \hline \end{array} & & \begin{array}{|c|} \hline \\ \hline \end{array} \\ \mathbf{A}^H & & \mathbf{b} \end{array}$$

Which gives us,

$$\begin{array}{ccc} \begin{array}{|c|} \hline \\ \hline \end{array} & \begin{array}{|c|} \hline \\ \hline \end{array} & \\ \mathbf{A}^H \mathbf{A} & \mathbf{x} & \end{array} = \begin{array}{ccc} \begin{array}{|c|} \hline \\ \hline \end{array} & & \\ & \mathbf{A}^H \mathbf{b} & \end{array}$$

Since $\mathbf{A}^H \mathbf{A}$ is a square matrix, \mathbf{x} can be found by equation 7.1.

$$\mathbf{x} = (\mathbf{A}^H \mathbf{A})^{-1} \mathbf{A}^H \mathbf{b}$$

The matrix $(\mathbf{A}^H \mathbf{A})^{-1} \mathbf{A}^H$ is called the Moore-Penrose pseudo inverse.

Applying this method to the previous example, we get

$$\begin{aligned} \begin{pmatrix} 3 & 4 \end{pmatrix} \begin{pmatrix} 3 \\ 4 \end{pmatrix} x &= \begin{pmatrix} 3 & 4 \end{pmatrix} \begin{pmatrix} 4 \\ 10 \end{pmatrix} \\ 25x &= 52 \\ x &= \frac{52}{25} \quad \square \end{aligned}$$

Which is indeed the same answer as we found in equation 7.4.

7.2 Python scripts

Listing 1: Code for computing the convergence plot for non-directional calibration

```
1 '''
2 Created on May 12, 2015
3
4 @author: nikki
5 '''
6
7 #Now we are going to make it work for N data sets per station. And add a plot of the
8   residue against the number of iterations.
9 #With faster method of calculating Z.
10 from __future__ import division
11 import numpy as np
12 import scipy.io as sc
13 import scipy.linalg as lin
14 import sys
15 from matplotlib import pyplot
16
17 def stefcal_nondir(M, R, tau=0.000001, i_max=100):
18     nst = len(R)
19     numsamples=M.shape[0]/M.shape[1]
20     print "numsamples: ", numsamples
21
22     #Initiate g[0]
23     gs = []
24     gs = gs + [np.ones(nst, dtype=complex)]
25
26     #Modify R to be in the same form as M:
27     R = np.tile(R, (numsamples, 1))
28     print R.shape
29
30     norms=[]
31
32     for i in range(1,i_max):
33
34         g = np.zeros(nst, dtype=complex)
35
36         for p in range(0,nst):
37
38             Zp= np.tile(gs[i-1],numsamples) * M[:, p]
39             g[p] = (R[:,p].conjugate().dot(Zp)).item() / lin.norm(Zp)**2
40
41         gs = gs + [g]
42         if np.mod(i, 2) == 0 and i > 0:
43
44             norm = lin.norm(gs[i] - gs[i-1]) / lin.norm(gs[i])
45
46             norms=norms+[norm]
47
48             if norm >= tau or norm <= -tau:
49
50                 gs[i] = (gs[i] + gs[i-1])/2
51
52             else:
53                 break
54
55     print "Norms: ", norms
56     print "Number of iterations:", i
57
58     I = np.arange(2, i+1, 2)
59     pyplot.plot(I, norms, color='green')
60     pyplot.yscale('log')
61     pyplot.ylabel('Residue')
62     pyplot.xlabel('Number of iterations')
63     pyplot.show()
64
65     return gs
```

```

66
67 #Loading the data from a matlab file
68 data = sc.loadmat('nikki.m')
69
70 #Model:
71 R = data[ 'Vm' ][0:10, 0:10]
72
73 #Data (this is just to simulate the real data, which will be in a similar format)
74 N = 6
75 M = np.tile(data[ 'V' ][0:10, 0:10], (N, 1))
76
77
78 gs=stefcal_nondir(M,R)
79
80 print gs

```

Listing 2: Code for computing the convergence plots for different number of directions (Figure 14).

```

1  '''
2  Created on May 12, 2015
3
4  @author: nikki
5  '''
6  #Making a directional stefCal. With the function adapted, and reduced data. With
7    variable amount of directions and multiple plots in one graph.
8  from __future__ import division
9  import numpy as np
10 import scipy.io as sc
11 import scipy.linalg as lin
12 from matplotlib import pyplot
13 import sys
14
15 Nrstations = 86
16 Nrfreq = 40
17 Nrdir = 5
18
19 def stefcal_dir(tau=0.000001, i_max=1000):
20     #Make a list for the different norms per direction
21     normlist = []
22
23     for d in range(1, Nrdir+1):
24         print "D IS", d
25         #----- Model -----
26         for b in range(0, d):
27             data = sc.loadmat('Patch_%s_%s.mat' % (b, d))
28             Cd = data[ 'Vmcube' ]
29             Cd = Cd[0:Nrstations, 0:Nrstations, :, 0:Nrfreq]
30             Cd = np.squeeze(Cd)
31             Cd = np.dsplit(Cd, Cd.shape[-1])
32             Cd = np.squeeze(Cd)
33             Cd = np.vstack(Cd)
34             Cd = np.matrix.conjugate(Cd)
35             #We want to make a 3d tensor, so we add to every matrix one empty dimension
36             Cd = np.expand_dims(Cd, axis=0)
37
38             if b == 0:
39                 C = Cd
40             else:
41                 C = np.concatenate((C, Cd), axis=0)
42
43         #----- Data -----
44         #Read the data from the file
45         data = sc.loadmat('Patch_0_%s.mat' % d)
46         V = data[ 'Vcube' ]
47         #Reduce the amount of data
48         V = V[0:Nrstations, 0:Nrstations, :, 0:Nrfreq]
49         # Get rid of the dimension of 1
50         V = np.squeeze(V)
51         #Split the frequency dimensions and save them in a list

```

```

52     V = np.dsplit(V, V.shape[-1])
53     #Get rid of the dimension of 1: now it is a list with arrays of dimension 2
54     V = np.squeeze(V)
55     #Stack all the arrays in the list beneath eachother
56     V = np.vstack(V)
57     #Because the dimensions got switched, we have to conjugate the matrix
58     V = np.matrix.conjugate(V)
59
60     nst = V.shape[1]
61     numsamples=V.shape[0]/V.shape[1]
62     dir = C.shape[0]
63
64     #Initiate G_oud
65     G_oud = np.ones((dir, nst),dtype=complex)
66
67     #Make an empty list for the norms
68     norms=[]
69
70     for i in range(1,i_max):
71         #Make a new, empty matrix for G_nieuw
72         G_nieuw = np.zeros((dir, nst),dtype=complex)
73         print i
74
75         for p in range(0,nst):
76             #Calculate the matrix A (= G * C)
77             A = np.tile(np.matrix.conjugate(G_oud).T, (numsamples, 1)) * C[:, :, p
78 ].T
79             #Fill the columns of G_nieuw with the solution of the least squares
80             method
81             G_nieuw[:, p] = np.linalg.lstsq(A, V[:, p], rcond=-1)[0]
82
83             if np.mod(i, 2) == 0 and i > 0:
84                 #Calculate the norm
85                 norm = lin.norm(G_nieuw - G_oud) / lin.norm(G_nieuw)
86                 #Add the norm to the list of norms
87                 norms=norms+[norm]
88
89                 G_nieuw = (G_nieuw + G_oud) / 2
90
91                 G_oud = G_nieuw
92                 print "NORMS", norms
93                 normlist.append(norms)
94
95                 print "normlist", normlist
96                 print "i is", i
97                 print "norms is", len(normlist[0])
98
99                 I = np.arange(2, i+1, 2)
100                 pyplot.plot(I, normlist[0])
101                 pyplot.plot(I, normlist[1])
102                 pyplot.plot(I, normlist[2])
103                 pyplot.plot(I, normlist[3])
104                 pyplot.plot(I, normlist[4])
105                 pyplot.ylim(ymax = 1, ymin = 10**(-5))
106                 pyplot.yscale('log')
107                 pyplot.xscale('log')
108                 pyplot.ylabel('Relative improvement (log)')
109                 pyplot.xlabel('Number of iterations (log)')
110                 pyplot.legend(['D = 1', 'D = 2', 'D = 3', 'D = 4', 'D = 5'], loc='lower left')
111                 pyplot.savefig('MultD_86x40_1000i.pdf', bbox_inches='tight')
112                 pyplot.show()
113
114             return G_nieuw
115
116 G=stefcal_dir()
117 print G

```